TU Delft

Delft University of Technology

On Safety in Machine Learning

Viering, T.J.

DOI
10.4233/uuid:a908aba1-8210-4b75-b69f-77aea7871a56

Publication date
2023

Document Version
Final published version

Citation (APA)
Viering, T. J. (2023). *On Safety in Machine Learning*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:a908aba1-8210-4b75-b69f-77aea7871a56

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

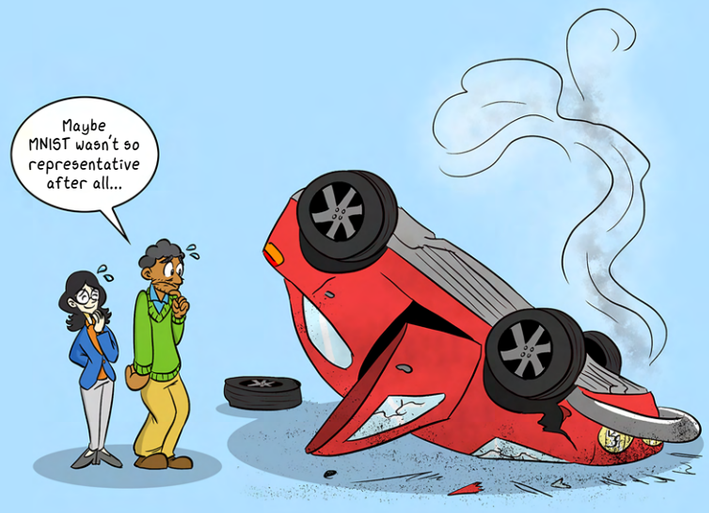
Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

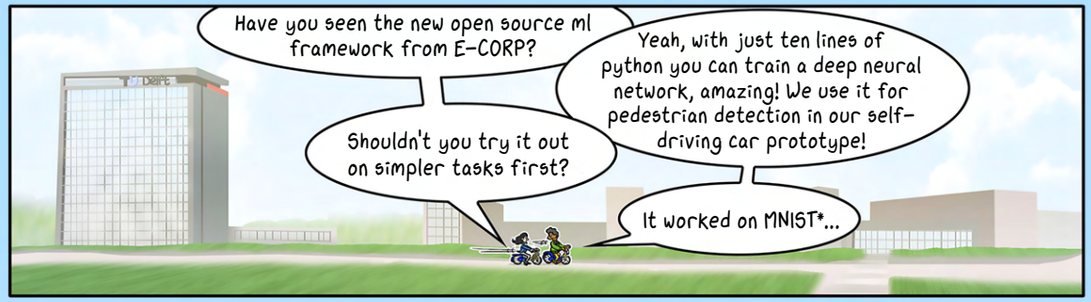
Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

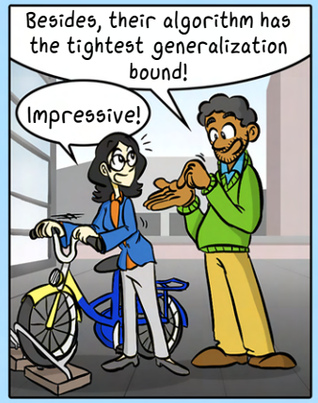
This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

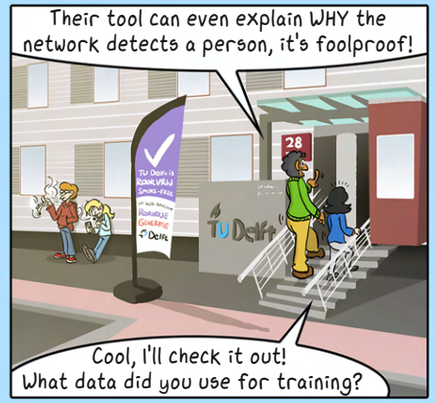
ON SAFETY IN MACHINE LEARNING

TOM JULIAN VIERING



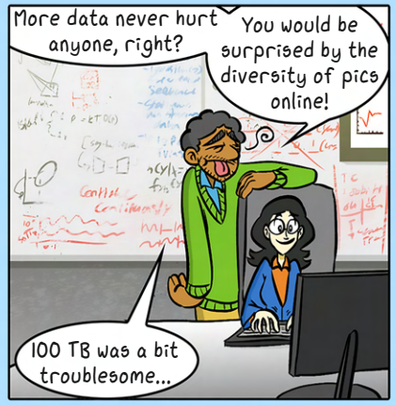


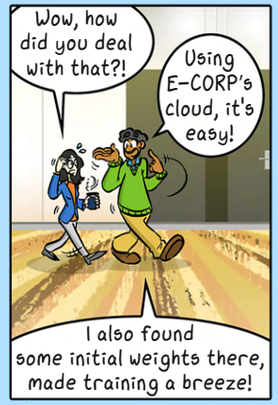




ON SAFETY IN MACHINE LEARNING

TOM JULIAN VIERING





*MNIST is a database of handwritten digits that is commonly used for benchmarking various machine learning algorithms

On Safety in Machine Learning

On Safety in Machine Learning

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Thursday, 4th of May, 2023 at 10:00.

by

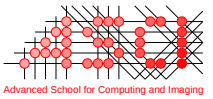
Tom Julian VIERING

Master of Science in Computer Science,
Delft University of Technology, The Netherlands,
born in Leiden, The Netherlands.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. E. Eisemann,	Delft University of Technology, *promotor*
Prof. dr. M. Loog,	Delft University of Technology, *promotor*
Independent members:	
Prof. dr. B. Hammer	Bielefeld University, Germany
Prof. dr. M. Biehl,	Rijksuniversiteit Groningen, The Netherlands
Prof. dr. M. Bicego,	University of Verona, Italy
Prof. dr. A.E. Zaidman	Delft University of Technology, The Netherlands
Dr. J.N. van Rijn	Leiden University, The Netherlands
Prof. dr. ir. M.J.T. Reinders	Delft University of Technology, *reserve member*



This work was carried out in the Advanced School for Computing and Imaging (ASCI).
ASCI dissertation series number 443.

Printed by: ProefschriftMaken.nl

Front & Back: Cover art by Jonathan van Engelenhoven a.k.a. banjoofjustice

Copyright © 2023 by T.J. Viering

ISBN 978-94-6469-259-4

An electronic copy of this dissertation is available at
`https://repository.tudelft.nl/`.

Voor Peter en Alma

"Evolution forged the entirety of sentient life on this planet using only one tool... The mistake."

Dr. Robert Ford (Westworld)

CONTENTS

SUMMARY

This dissertation focuses on safety in machine learning. Our adopted safety notion is related to robustness of learning algorithms. Related to this concept, we touch upon three topics: explainability, active learning and learning curves.

Complex models can often achieve better performance compared to simpler ones. Such larger models are more like blackboxes, whose inner workings are much harder to understand. However, explanations for their decisions may be required by law when these models are used, and may help us further improve them. For image data and CNNs, Grad-CAM produces explanations in the form of a heatmap. We construct CNNs whose heatmaps are manipulated, but whose predictions remain accurate, illustrating that Grad-CAM may not be robust enough for high stakes tasks such as self-driving cars.

Machine learning often require large amounts of data for learning. Data annotation is often expensive or difficult. Active learning aims to reduce labeling costs by selecting data in a smart way — instead of the default, random sampling. Active learning algorithms aim to find the most useful samples. Surprisingly, we find that active learning algorithms with strictly better performance guarantees perform worse empirically. The cause: their worst-case analysis is unrealistic. A more optimistic average-case analysis does explain our empirical results. Thus better guarantees do not always translate to better performance.

A learning curve visualizes the expected performance versus the sample size a learning algorithm is trained on. These curves are important for various applications, such as estimating the amount of data needed for learning. The conventional wisdom is that more data equals better performance. This means a learning curve strictly improves with more data, or in other words, is monotone. Deviations can surely be explained away by noise, chance, or a faulty experimental setup?

To many in our field this may come as a surprise, but this behavior cannot be explained away. We survey the literature and highlight various non-monotone behaviors, even in cases where the learner uses a correct model. Our survey finds that learning curves can have a variety of shapes, such as power laws or exponentials, but there is no consensus and a complete characterization remains an open problem. We also find simple learning problems in classification and regression that show new non-monotone behaviors. Our problems can be tuned so non-monotonicity occurs for any sample size.

Is there a universal solution to make learners montone? We design a wrapper algorithm that only adopt a new model if its performance is significantly better on validation data. We prove that the learning curve of the wrapper is monotone with a certain probability. This provides a first step towards safe learners that are guaranteed to improve with more data. Many questions regarding safety remain, however, this thesis may provide inspiration to develop more robust learning algorithms.

The main take-aways are (TLDR):

- Strictly tighter generalization bounds do not imply better performance.
- Explanations provided by Grad-CAM can be misleading.
- Even in simple settings more data can lead to worse performance.
- We provide ideas to construct learners that always improve with more data.

SAMENVATTING

Dit proefschrift richt zich op veiligheid van zelflerende algoritmes (machine learning). De precieze definitie van veiligheid die we hanteren heeft te maken met de robuustheid van zelflerende algoritmen. Dit proefschrift beslaat drie themas: de uitlegbaarheid van leeralgoritmen, actief leren (active learning) en leercurven (learning curves).

Complexe modellen presteren vaak beter dan simpelere als ze op voldoende data zijn getraind. Als we zulke complexe modellen willen toepassen in de praktijk, vereist de wet vaak van ons dat we de beslissingen van deze modellen kunnen uitleggen. Bovendien, als we kunnen begrijpen waarom complexe modellen bepaalde beslissingen maken, kunnen we ze ook gemakkelijker verbeteren. Dit is de motivatie om leeralgoritmen uit te leggen.

Voor visuele data en convolutionele neurale netwerken is de uitleg vaak ook visueel. De populaire methode Grad-CAM produceert een heatmap: een beeld waarin de intensiteit van elke pixel aangeeft hoe belangrijk deze pixel was bij de beslissing van het neurale netwerk. In dit proefschrift laten we zien dat we neurale netwerken kunnen bouwen die goed presteren, maar die tegelijkertijd ook de uitleg van Grad-CAM misleidt. Dit illustreert dat Grad-CAM niet robuust is. Voor toepassingen waarin veiligheid hoog in het vaandel staat, zoals bij zelfrijdende autos, kunnen we dus beter niet vertrouwen op Grad-CAM.

Voor het leren van complexe algoritmes is vaak een grote hoeveelheid geannoteerde data nodig. Die annotaties zijn vaak kostbaar en niet gemakkelijk verkrijgbaar. Actief leren heeft daarom als doel om de hoeveelheid data die nodig is om te leren te verminderen. Dit wordt gedaan door een actief leer algoritme. Dit algoritme bepaalt welke data gelabeld moet worden. Door hierin slimme keuzes te maken is er minder data nodig om te leren.

In dit proefschrift laten we zien dat algoritmen voor actief leren met strikt betere theoretische garanties in de praktijk slechter presteren. We laten zien dat dit komt door het feit dat de theoretische analyse gebaseerd is op een worst-case analyse die onrealistisch is. Door een theoretische analyse die gebaseerd is op een realistischer scenario kunnen we wel onze resultaten verklaren. Kortom; betere theoretische garanties vertalen zich niet altijd naar betere empirische resultaten.

Tenslotte bestuderen we leercurves. Een leercurve visualiseert de gemiddelde prestaties van een leeralgoritme als functie van de hoeveelheid data die is gebruikt voor het leren. Deze curves kunnen bijvoorbeeld worden gebruikt om te schatten hoeveel data nodig is om een bepaalde prestatie te behalen. De gangbare gedachte is dat meer data een betere prestatie impliceert. Oftewel: de leercurve is monotoon. Afwijkingen hiervan kunnen natuurlijk worden verklaard door een gebrekkige experiment, of toeval?

Voor velen in ons vakgebied zal dit als een verassing komen: deze resultaten waar leren slechter wordt met meer data zijn gebaseerd op valide simulaties. We doen een uitgebreide analyse van de literatuur en laten zien dat er meerdere situaties zijn waar leeralgoritmes niet-monotone leercurves vertonen. Dit kan zelfs gebeuren als het leeralgoritme correcte modelaannames maakt. In ons literatuur onderzoek vinden we leercurves met een breed sca-

la aan verschillende gedragingen, zoals polynomiaal of exponentieel verval. In de literatuur is er geen consensus over de definitieve vorm van leercurves en een volledige karakterisatie van alle leercurven ontbreekt. In ons onderzoek vinden we zelfs een aantal nieuwe simpele classificatie en regressie problemen waar leeralgoritme nieuw niet-monotoon gedrag laten zien. Ons leerprobleem kan zelfs zo gebouwd worden, dat de leercurve verslechterd voor elk willekeurig gekozen grootte van de training set.

Is er een universele methode om leeralgoritmen monotoon te maken? Dat is de laatste vraag die dit proefschrift onderzoekt. We bouwen een algoritme dat gecombineerd kan worden met elk leeralgoritme dat een poging doet het leeralgoritme zich beter te laten gedragen. Dit algoritme monitort continue de prestaties van het leeralgoritme, en grijpt in als het model slechter wordt. In dat geval wordt het vorige beste model gehanteerd. We bewijzen dat dit resulterende algoritme monotoon is met een bepaalde kans. We geloven dat dit een eerste stap vormt in de richting van veilige leeralgoritmen die altijd verbeteren met meer data. Er blijven veel vragen rondom veiligheid van zelflerende algoritmen in de context van robuustheid, maar dit proefschrift geeft hopelijk inspiratie tot het bouwen van robuustere leeralgoritmes.

Mocht deze samenvatting al aan de lange kant zijn, dan hierbij een nog beknoptere samenvatting (TLDR):

- Strikt betere theoretische garanties garanderen geen betere empirische prestaties.
- Heatmaps gegenereerd door Grad-CAM kunnen misleidend zijn.
- Zelfs in simpele gevallen kan meer data leiden tot slechtere prestaties.
- We nemen een eerste stap in de richting van het bouwen van leeralgoritmen die altijd beter presteren als ze meer data voorgeschoteld krijgen.

1

INTRODUCTION

The most exciting phrase in science is not "Eureka!" but "That's funny..."

Isaac Asimov

Machine learning as a technology is not safe. This is perhaps best illustrated by the fact that authors for the NeurIPS conference are now asked to write a statement that considers the ethics and societal impact of their manuscript [1]. In this case, safety relates to dangers to society posed by automated systems constructed using machine learning, and can be related to algorithmic discrimination, classification of race, automated weapons, etc. In safety-critical applications, such as self-driving cars, mistakes of the machine can also result in physical harm [2].

Whereas the previous discusses safety in the context of applications, we are concerned with safety in machine learning theory. In that case, one may expect that less can go wrong, but even there it is hard to exclude undesirable outcomes. The studied safety topics in the theory literature are diverse [3–5]. Safety can relate to reinforcement learning in several ways, as these algorithms may be used to control robots in the real world. For example, when these algorithms are exploring their environment, they may inadvertently perform unsafe actions, or errors in their objective function may lead them to do so. Safety can also relate to robustness of machine learning algorithms [5, 6], which is our focus.

A well-known example of safety that is studied in theory are adversarial attacks. In that case inputs to the machine learning system are manipulated by an adversary to cause it to make wrong decisions [7]. Closely related topics are the monitoring of machine learning models, which can be automatic by another system or by humans. The latter has connections with model interpretability, since humans need to understand when the system malfunctions. In the context of monitoring, a popular research direction aims to detect distributional shifts or other forms of attacks and how to defended against them [8].

We study robustness and safety in the context of explainability, active learning and learning curves. Since we deal with supervised learning and its variants, we first recap that setting. For interested readers, we also provide some more background information on learning theory in the Frequentist framework, such as Empirical Risk Minimization (ERM)

1

and Probably Approximately Correct (PAC) learning, and finally we provide a very brief primer on Bayesian machine learning. Afterward, we discuss our definition of safety, and we review some related work that studied this notion of safety in semi-supervised learning and domain adaptation. Then, we introduce the three topics in more detail, and we wrap up the introduction with an overview of the chapters, the most important notation used in this dissertation and the most important terminology.

1.1. PRELIMINARIES 1: SUPERVISED LEARNING

In supervised learning, we assume there is an unknown distribution $P_{\mathcal{X}\mathcal{Y}}$ over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ are the features or covariates and $\mathcal{Y}$ are the labels or outputs that we are interested in predicting. We usually assume that the features live in a d-dimensional space: $\mathcal{X} \in \mathbb{R}^d$. Standard settings are binary classification where $\mathcal{Y} \in \{+1, -1\}$ corresponding to positive and negative class labels and in regression $\mathcal{Y} \in \mathbb{R}$ corresponding to the regression targets.

In regression, there often will be an underlying regression function $f : \mathcal{X} \to \mathcal{Y}$ that we are trying to learn. Observations are typically noisy, and thus we have that $y = f(x) + \epsilon$, where ϵ is, for example, i.i.d. Gaussian noise with zero mean and a variance of σ^2. Furthermore, we assume that there is a marginal distribution $P_{\mathcal{X}}$, this describes the distribution of the features or covariates. It is obtained by marginalizing out the $\mathcal{Y}$ out of $P_{\mathcal{X}\mathcal{Y}}$. The underlying regression function, the noise and $P_{\mathcal{X}}$ completely determine $P_{\mathcal{X}\mathcal{Y}}$ in that case.

In binary classification, we can consider an underlying decision function, $\eta : \mathcal{X} \to [0, 1]$, that is defined as $\eta(x) = P(y = 1|x)$, the posterior probability of the positive label. Note that, if we fix x, and sample a label y, the label y behaves as a Bernoulli-variable with success probability $\eta(x)$. Thus, a binary classification problem $P_{\mathcal{X}\mathcal{Y}}$ can equivalently be characterized by $P_{\mathcal{X}}$ and $\eta(x)$.

We receive n independent and identically distributed (i.i.d.) samples from $P_{\mathcal{X}\mathcal{Y}}$. This sample is called the training set, which we indicate by S_n. We call such samples labeled, as for each object in the sample the label or target is available. A learner, which we indicate by A, uses the training set to construct a model, h. This model takes as input $x \in \mathcal{X}$ and aims to predict the corresponding $y \in \mathcal{Y}$. All possible models h that A can output form the hypothesis class H. For example, one may consider the class of all linear models, $H = \{x \to w^T x + b : w \in \mathbb{R}^d, b \in \mathbb{R}\}$. We write predictions of a model as $\widehat{y}$.

The performance of the model is measured using some loss function $l(\widehat{y}, y)$. For classification often the zero-one loss is used: $l_{01}(\widehat{y}, y) = 1_{\widehat{y} \neq y}$. This loss function counts the number of mistakes, and thus when averaging this loss over samples, we obtain an estimate of the error rate of this classifier, which is defined as the probability of making a mistake. For regression the squared loss is typically used: $l_{\text{sq}}(\widehat{y}, y) = (\widehat{y} - y)^2$. The goal for the learner is to return a model with good generalization performance; meaning that the loss on unseen objects is small. The generalization performance of a model is measured by the risk,

$$R(h) = \mathbb{E}_{(x,y)\sim P_{\mathcal{X}\mathcal{Y}}} l(h(x), y). \tag{1.1}$$

The risk for classification can be interpreted as the probability of making a classification error on new and unseen data sampled from $P_{\mathcal{X}\mathcal{Y}}$, or in other words, this is equal to the error rate.

Let $A(S_n)$ be the model returned by the learner when trained on the training set S_n. Because the training set is a random sample, the risk of the learner, $R(A(S_n))$ is a random variable. To obtain a single number to characterize the performance of A, we take the expectation and obtain the expected risk:

$$R(A) = \mathbb{E}_{S_n \sim P} R(A(S_n)). \tag{1.2}$$

Often, it is logical to compare the performance of our models to the best performance that is attainable in theory. To get to such models, we assume we have access to the (typically unknown) distribution $P_{\mathscr{X}\mathscr{Y}}$. For the regression case with the mean squared error, we can never perform better than the underlying regression function f. Thus the best possible performance is given by $R(f)$. For classification with the zero-one loss, it is optimal to predict class $+1$ if $\eta(x) = P(y = 1|x) > \frac{1}{2}$, otherwise one should predict -1. This decision rule is called the Bayes-optimal classifier or Bayes-classifier and we indicate it by h_η. When $\eta(x) = \frac{1}{2}$, this indicates the Bayes decision boundary (where the optimal predictions go from one class to the other). The performance of the Bayes-classifier is referred to as the Bayes error, which we indicate by $R(h_\eta)$.

However, our learners may never be able to reach the minimal error $R(h_\eta)$ (classification) or $R(f)$ (regression). This is because our learners can only return models from our hypothesis class H. For example, the optimal decision boundary may be given be a quadratic function. In case we use linear classifiers as hypothesis class, we will surely incur some loss in performance due to the approximation made by the hypothesis class. Thus, for this reason, we will also often compare our performance to the best model h^* in our class: $h^* = \operatorname{argmin}_{h \in H} R(h)$.

This motivates us to decompose the error of a model as:

$$R(h) = \underbrace{R(h) - R(h^*)}_{\text{estimation error}} + \underbrace{R(h^*) - R(h_\eta)}_{\text{approximation error}} + \underbrace{R(h_\eta)}_{\text{irreducible error}} \tag{1.3}$$

The first term is the estimation error or excess risk. This error can be attributed to the learning algorithm, it will be large if we get a very unrepresentative training set, or if the learner overfits. Thus this error is large due to estimating the wrong function h due to the finite sample we have access to. The second part is the approximation error due to using a limited hypothesis set H. In case the approximation error is zero, then a learning problem is called realizeable. If this error is non-zero, the setting is called agnostic. The final term is the irreducible error, this error can never be reduced due to the inherent noise of the learning problem, and this coincides exactly with the Bayes error. Note that in case of regression, this should be $R(f)$ instead of $R(h_\eta)$.

1.2. PRELIMINARIES 2: ERMS AND PAC LEARNING

Now we will discuss some basics of machine learning theory, following the excellent textbook of Shalev-Shwartz and Ben-David [9]. In this dissertation we often consider Empirical Risk Minimizers (ERMs). The empirical risk is given by

$$\widehat{R}(h) = \frac{1}{n} \sum_i l(h(x_i), y_i) \tag{1.4}$$

which is the empirical loss on the training set S_n. An empirical risk minimizer returns the hypothesis that minimizes $\widehat{R}(h)$. For regression this typically yields a tractable optimization problem because the mean squared error is smooth and convex. However, for classification, this approach is usually unfeasible, as the zero-one loss is difficult to optimize in the agnostic case [9, page 105-119].

Therefore, for classification, we often use a different loss for learning, also referred to as surrogate loss. In this dissertation we will often take the squared loss as surrogate loss for learning a classifier. We refer to the resulting classifier as a least squares classifier or Fisher classifier (see [10, p. 186] for details regarding the origins of the name and its relation to the Fisher criterion). If we have $n > d$, the least squares problem has a unique solution, and this is generally referred to as underparameterization (less parameters than samples).

The overparamatrized setting is the case where we have more parameters than training samples, $d > n$. Then the least squares solution on the training set is not unique and there are many solutions. In that case we will take the solution where the parameters have minimum euclidean norm (in case of linear classifiers; we defer a discussion of the kernelized case to Chapter 3). This solution is found by solving the regression with a pseudo-inverse. This model is called Pseudo-Fisher by Duin [11] or Minimum Norm Linear Regression (MNLR) by Loog *et al.* [12]. Note that, when Pseudo-Fisher is used for the case $n > d$, it coincides with the Fisher classifier.

The estimation error (from Equation 1.3) can be analyzed using generalization bounds or closely related learnability notions. One such notion is that of Probably Approximately Correct (PAC) learning for the case of classification with the zero-one loss. The PAC framework analyzes learners that train on i.i.d. data of size n. Using the results of PAC, one can show that for empirical risk minimizers an upperbound holds with probability $1-\delta$ on the excess risk for all distributions (learning problems) $P_{\mathscr{X}\mathscr{Y}}$. The parameter $1-\delta$ indicates the probability of successful learning (hence, probably), and $1-\delta$ is also called the "confidence". Because the training set is a random sample, it is unavoidable that we may get a very unrepresentative training set, and therefore a probability of failure is inevitable. The upperbound is of the form $R(h) - R(h^*) \leq \frac{C}{n}$ (realizeable case) and $R(h) - R(h^*) \leq \frac{C'}{\sqrt{n}}$ (agnostic case). In case a finite hypothesis class is used, C and C' depends on the size of the class and δ. Such an upper bound is called a generalization bound. Note that for PAC bounds, the upperbound cannot depend on $P_{\mathscr{X}\mathscr{Y}}$.

The PAC analysis can also be extended to classes of infinite size, such as the class of all linear classifiers. For infinite classes, there is a combinatorial property of the hypothesis class H, called the VC-dimension in the bounds. For a precise definition and concise introduction see [9, Chapter 6]. Roughly speaking, the more complex the hypothesis class, the larger the VC-dimension.

If a class is too complex, e.g. it can fit anything, the VC-dimension is infinite and PAC-learning is impossible with that hypothesis class. This fundamental result is called the No-Free-Lunch theorem (see [9, Chapter 5.1] for more detail). From the No-Free-Lunch theorem it follows that for such a hypothesis class, no learning algorithm exists for which such an upperbound on the excess risk (that in the limit goes to zero) can be given that holds uniformly for all distributions $P_{\mathscr{X}\mathscr{Y}}$. The No-Free-Lunch theorem basically shows that for each learning algorithm, a distribution can be found on which the learner fails. With other words, learners with such an hypothesis class can overfit, since the training error

is zero, while the error on the distribution can be quite large. This fundamental result can be interpreted as that such a hypothesis class that is too flexible lacks prior knowledge. The hypothesis class needs to be made less flexible, otherwise successful learning in the PAC-sense is not possible.

Besides PAC-learning, there are also other learnability concepts. One such learnability concept is called Structural Risk Minimization (SRM). Structural risk minimization recognizes, like PAC learning, that our hypothesis set should not be too complex, otherwise we overfit. But on the other hand, the hypothesis class should not be too simple either, otherwise the approximation error will be very large. In PAC learning, the hypothesis class H is fixed, while Structural Risk Minimization aims to choose a class that achieves a small approximation and estimation error.

Instead of choosing an appropriate hypothesis class as in SRM, we can also directly minimize the empirical risk and the complexity of our hypothesis simultaneously: this way a trade-off can be made between a model that is not too complex and provides a good fit. For linear classifiers, we can measure the complexity of the model by the norm of the vector of the parameters. This results in a different kind of learner, a Regularized Loss Minimization rule:

$$\operatorname*{argmin}_h \widehat{R}(h) + \mu ||w||^2 \tag{1.5}$$

The first term is the empirical risk that we have seen before. The second term forces the model h to be simpler; for the simplest model $w = 0$ this term is minimal. This term is called the regularization term. Here we have chosen L_2 regularization (since we use the 2-norm), but other norms can also be used for learning. In this objective μ is a hyperparameter that controls the trade-off between a low empirical risk and low complexity. In this dissertation, we will see a learner of this type with the squared loss, which we call Regularized Least Squares. If such learners minimize a smooth loss (such as the squared loss) and a convex hypothesis set is used, techniques can also be used to come to generalization bounds for such learners using arguments based on stability [9, Chapter 13] or using generalization bounds based on Rademacher complexity [9, Chapter 26]. Latter bounds are more refined than VC-bounds, and can be used for general losses besides the zero-one loss.

The PAC learning framework and other generalization bounds provide a lot of insight into the problem of learning and impossibilities of learning. However, these framework based on bounds also have their limits and should be carefully interpreted. For example, most theoretical works show there is an upper bound on the excess risk under particular assumptions that decreases monotonically as a power law in terms of the size of the training set, $\frac{C}{n}$ (realizeable case) and $\frac{C'}{\sqrt{n}}$ (agnostic case). However, later on in this introduction, we will highlight that the actual excess risk may display different behavior as a function of n.

1.3. PRELIMINARIES 3: BAYESIAN MACHINE LEARNING

So far, we have discussed the Frequentist view of machine learning. It is a particular school of thought within statistics. An alternative to the Frequentist framework is offered by the Bayesian approach to statistics [13]. The advantage of the Bayesian approach is that it is useful if data arrives sequentially, and if we want to encode prior knowledge in our statistical models. Advantages of the Frequentist school are that no choices of prior are necessary,

making Frequentists statistics "more objective" in some sense. Frequentists typically use confidence intervals and p-values, while Bayesians use credible intervals and Bayes factors. Another key difference is that Frequentists machine learning algorithms will return a single model, which makes point predictions. In contrary, Bayesian machine learning algorithms return a whole probability distribution over models, where the probability is indicative of the belief in each model. Similarly, predictions are a whole distribution over values, each weighted by a probability indicating the amount of belief in their occurence.

One crucial Frequentist assumptions is that the true regression model f exists and is unknown and fixed. Or equivalently, for classification, that h_η is uknown and fixed. The second typical assumption is that data is generated from a distribution $P_{\mathcal{X}\mathcal{Y}}$. Interestingly, Bayesians take exactly the opposite assumptions: the data is fixed (non-stochastic!), and the model parameter f is a random variable.

To make the discussion more concrete, let us explain the example of Bayesian linear regression. A Bayesian, someone adheres to the Bayesian school of statistics, assumes $y = w^T x + \epsilon$, with ϵ i.i.d. Gaussian noise with mean zero and variance σ^2. This noise model is called the Gaussian likelihood. The Bayesian assumes that the variables x are given and fixed. He further assumes a prior on w, for example, $w \sim N(0, \Sigma_p)$, where Σ_p is a positive definite matrix of size $d \times d$. A common choice can be the identity matrix. The prior indicates the prior belief of the Bayesian which models may be suitable for the task, and is chosen *before* observing data for the particular learning problem.

Now, if the Bayesian wants to make predictions, he will first use Bayes rule to compute the posterior of w given the data of the training set. Let X be the $n \times d$ datamatrix of the training set and let Y be the n-dimensional vector of labels of the training set. Then Bayes rule applied to w gives the posterior distribution:

$$p(w|Y, X) = \frac{p(Y|X, w)p(w)}{p(Y|X)} \tag{1.6}$$

Here, $p(w)$ indicates the prior, $p(Y|X, w)$ is called the likelihood, and $p(Y|X)$ is the marginal. Assuming the noise ϵ is i.i.d., we have that $p(Y|X, w) = N(Xw, \sigma I_d)$, where I_n is the n-dimensional identity matrix. The marginal is given by integrating out w, thus:

$$p(Y|X) = \int_w p(Y|X, w)p(w)dw \tag{1.7}$$

Computing the posterior can be very challenging, but for this simple regression case with Gaussian noise and Gaussian prior, it turns out the posterior is also a Gaussian.

To make a prediction for a test point, we do a majority vote over all possible models w, where their vote is weighted by their posterior probability. Let us say we have a test point x' whose regression target we wish to predict. Our prediction is a random variable, let us call it $b = w^T x'$ (the randomness is stemming from the fact that w is a random variable). We can compute the mean of the variable under the posterior to get our most likely guess:

$$\int w^T x' p(w|X, Y)dw \tag{1.8}$$

Now we can see the key advantage of this Bayesian approach to machine learning. Because we take a majority vote over many models, we are less prone to overfitting, because the

majority vote spreads the belief over multiple models. Second, we can get an indication of the belief in our prediction. If the posterior is very concentrated on one model, the distribution $w^T x' p(w|X, Y)$ will be peaked. If many models explain the training set well, the posterior will be much broader, and then $w^T x' p(w|X, Y)$ will be broad and uncertain. So we can assess the certainty of our model by, for example, computing the variance of b under the posterior.

Note that, the most likely prediction for the test point for this simple model, is exactly the same as only using a single model w of the mean / mode of the posterior of w. However, for more complex probabilistic models this will generally not be the case. So while a Frequentist machine learner will estimate a point prediction for a new object, e.g. $w^T x$, instead a Bayesian will always predict a whole distribution of likely targets for a new object. This distribution provides not only the most likely estimate, but also provides information regarding the certainty of the estimates.

Just like in Frequentist learning theory, in Bayesian learning theory there are also common settings and assumptions. Some of the theoretical results in the Bayesian setting assume that the model is well-specified, or that there is no model mismatch. This means that the likelihood function, noise model, and prior are correct. Or in other words, that data is truly generated from a model from the prior. If this is not the case, the model is misspecified.

The notion of risk also changes. In the Frequentist setting there was one fixed groundtruth model. In the Bayesian setting, this cannot be the case, because the model is a random variable. In fact, in the Bayesian case, we consider a whole range of learning problems, $\mathscr{P}$. Because each w defines a new single learning problem $P_{\mathscr{X}\mathscr{Y}}(w)$, where we indicate the dependence of the learning problem on w. Thus, to compute the risk, we have to take another expectation with respect to w. We have to redefine the risks in that case to indicate the dependence on w.

$$R(h, w) = \mathbb{E}_{(x,y)\sim P_{\mathscr{X}\mathscr{Y}}(w)} l(h(x), y). \tag{1.9}$$

$$R(A, w) = \mathbb{E}_{S_n \sim P} R(A(S_n), w). \tag{1.10}$$

Now we can define the Problem Average Risk as:

$$R_{PA}(A) = \mathbb{E}_{w\sim P_W} R(A, w). \tag{1.11}$$

We call this a problem average, since we are integrating over multiple learning problems. The learning problems are now defined by the prior distribution P_W. However, we could equivalently define a distribution $\mathscr{P}$ over distributions $P_{\mathscr{X}\mathscr{Y}}$.

1.4. SAFETY AND PREVIOUS WORK ON SAFETY

We study robustness and safety in the context of explainability, active learning and learning curves. Let us now discuss our definition of safety, and we review some related work that studies this notion.

Our notion of safety compares the expected risk for a particular n of two algorithms A and B. Here B is a baseline that we always want to beat, e.g. we always want $R(A) \leq R(B)$. If this inequality holds, we call A safe. Note that this definition is similar to the statistical notion of dominance [13, p. 196]. The difference is that in the context of safety, we only

care about dominating a specific baseline B, while the notion of dominance means that A is better than a whole collection of B's. In safety this baseline B is natural or logical to beat, which depends on the setting. Let us give two examples to illustrate.

In semi-supervised learning, besides a labeled sample, the learner has also access to an unlabeled sample from the marginal distribution $P_{\mathscr{X}}$. Here the marginal distribution is obtained by marginalizing out the target y. For the unlabeled data we do not know the label of the objects. However, in many settings such unlabeled data is very easy to obtain. For example, in the context of images, it is quite straightforward to collect a large database of images from the internet that are unlabeled. Obtaining accurate labels (e.g. the object in the image) is much more costly and difficult. Therefore, it is logical to try and use the unlabeled data somehow in the learning algorithm, as it may give the learner more information. Thus, in semi-supervised learning, the goal is to exploit knowledge of the unlabeled sample to obtain better performance. In this case, it is obvious that a natural baseline B with which we want to compare is supervised learning, which does not use unlabeled data at all. After all, unlabeled data is supposed to boost the performance.

In domain adaptation, there are two distributions $P_{\mathscr{X}\mathscr{Y}}$, a source distribution and a target distribution, that model data coming from two different domains. Usually, the source and target domains are related. For example, source data could correspond to an image classification problem where data is collected with a webcam, while target data is the same classification problem only data is collected with a DSLR camera. The goal in this setting is to perform well on the target domain, thus the risk is measured on the target. Meanwhile usually little data from that domain is available. Instead, we have a lot of source data available, which is why we want to exploit the source data for learning. For the example of the webcam and the DSLR camera, if the pictures are captured in the same manner (e.g. same environment, poses, etc.), it seems likely that indeed the source data should be useful for learning of the target task. More formally, we will have that source and target distributions over $X \times Y$ will be approximately be the same, but the marginal distributions of the domains will differ (because of the different modalities). In domain adaptation the central question is how to efficiently combine data from source and target to get good performance on the target. In unsupervised domain adaptation, a more specialized setting, we receive labeled data from the source domain and unlabeled data from the target domain. The goal is to exploit the unlabeled data to adapt the classifier on the source data to perform well on the target domain. Here, supervised learning on the source data also provides a natural baseline, as this approach does not use any of the unlabeled target data and is not aware of the adaptation problem.

Initially sensible procedures or intuitive algorithms may turn out to be unsafe for both of these settings. As an example, self-learning [14] in semi-supervised learning seems straightforward. It works as follows. A supervised learner is trained on the labeled data, and the resulting model is used to predict the targets of the unlabeled data. These predictions are used as labels for the unlabeled data in the next iteration. The model is then trained again, now on unlabeled data and labeled data, and again the labels of the unlabeled data are predicted. This is repeated until the procedure converges.

However, examples are known where the performance of self-learning can be worse compared to supervised learning [15]. The provided insights by studying safety can therefore be surprising, since even basic procedures may turn out to be less safe than initially thought.

This can be surprising if empirical evidence does suggest good performance. This must mean that there are assumptions that we may not be aware of. Similarly, in domain adaptation, it turns out strong assumptions on the relation between the source and target distribution are necessary to ensure success [16]. Uncovering these assumptions will inform us which algorithms are to be preferred in what case. Therefore we think safety is worthwhile to investigate.

Note that we can draw some parallels between safety and the time or space complexity analysis of algorithms. For example the running time of algorithms is often studied in terms of big-O-notation. For example, a sorting algorithm may admit a $O(n^2)$ time complexity, where n refers to the items to be sorted. This usually refers to the asymptotic performance for large n. Meanwhile, in learning theory we are usually concerned with bounding the performance even for finite n (such as the PAC result). This is why we also focus on finite n for safety.

Both algorithmic complexity and learning theory however do not study the direct performance difference between algorithms, e.g. both bounds the performance of *individual* algorithms, such as $R(A) \leq \frac{C}{n}$ (note that the bound on the excess risk also directly implies a bound on the risk itself). In safety, however, we directly look at their difference $R(A) - R(B)$ for *the same learning problem*. The conclusions drawn are significantly different. For example, a worst-case bound of A could be tighter than that of B, but this says nothing about their relatively performance for all cases, as such, it could turn out that A performs worse for the majority of all cases compared to B. Meanwhile, if A is safe, it means it will perform better than B in all cases. Thus, safety guarantees are stronger.

For both semi-supervised learning and domain adaptation strict safety guarantees have been derived. Strict safety means that A is always better than B (non-strict safety would imply A and B may sometimes tie). These proofs have been derived in the transductive setting, where performance is not measured using the risk, but the loss is averaged over a fixed and known sample which is available to the learner, except for the labels. Strictly safe semi-supervised learning and domain adaptation is possible for some learners and some loss functions [17–19]. For the hinge loss, the loss function underlying the support vector machine, however, it turns out that strictly safe semi-supervised learning is generally not possible [20]. Weaker safety guarantees may then still be possible, for example, Biggio *et al.* [7] provide safety guarantees under the assumption that the true model lies in a low density region.

We touch upon three topics in the context of safety: explainability, active learning, and learning curves. The first two topics will only come back in two individual chapters, while the topic of learning curves is studied more extensively in three chapters. We now introduce the topics in more detail, and then we end with a overview of the chapters, the most important notation in this dissertation and a list of the most important terminology (glossary). Note that the first topic, explainability, does not follow the exact definition of mathematical safety as defined before and should be considered in a broader view regarding robustness.

1.5. EXPLAINABILITY

The first topic focuses on explainability of machine learning models. Explainable AI has as goal to improve our understanding of why a complex machine learning model produces

particular outputs [21]. We know the the equations and parameters that govern the models behavior, but due to their complexity the whole can be difficult to interpret and understand. Thus, while the weight matrices and model equations constitute a mathematically correct explanation, it can fall short in other respects of what constitutes a good explanation.

The definition of a good explanation is difficult, but what should be clear is that it depends on the intended use of the explanation [21]. For example, if a user has their credit application automatically rejected and the system needs to explain why, the weight matrix of a model is insufficient. In such an application the explanation needs to be in language the user can understand, for example: "Your request for this loan was rejected because the requested amount was too high". However, for a designer of the machine learning system, such explanations are much too simplified.

Explanations are useful to get a better understanding of our models. For example, was the decision fair [21]? In particular for machine learning models in high-stakes settings this is important, such as when a judge in the United States relies on the judgment of a machine learning system to assess whether a suspect will re-offend [22]. In the European Union, the General Data Protection Regulation (GDPR), a recent regulation protecting privacy and data of European citizens, includes articles that come down to the right to an explanation [23]. This law says that, whenever people are automatically judged in some way by algorithms, they have a right to an explanation of the automatic decision. Besides this law, safety-critical applications such as autonomous vehicles will likely also require explanations of automated decisions, so questions such as blame can be resolved. Besides motivations from applications, machine learners can use explanations to obtain further insight in why their model makes particular mistakes and how models may be improved [21].

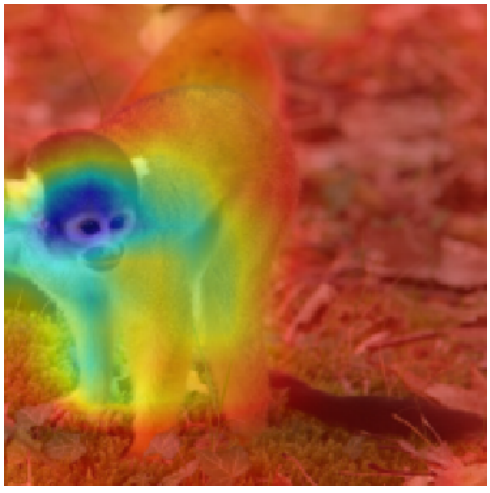


Figure 1.1.: Heatmap that explains the classification output 'monkey'.

Regarding explanations we focus on image data and classification[1], where a popular form of explanation of a single classification outcome is a heatmap, see Figure 1.1 for an example.

[1]The rest of the thesis does not consider only image data, specifically, but considers all forms of data.

The heatmap assigns each pixel in the input a value, indicating the importance of that pixel in the final classification. For this type of explanation it can be made mathematically precise what constitutes a good explanation. For example, one objective metric is characterized by the fact that blacking out important pixels should change the outputs of the model more rapidly than if non-important pixels are removed [24].

Many explanation methods have been proposed to built such heatmaps. One popular technique to build explanations for convolutional neural networks is called Grad-CAM [25]. This technique performs a backward pass through a neural network to determine the sensitivity of intermediate values on the output. These sensitivities have associated spatial locations that can be used to build such heatmaps.

Perhaps its simplicity has led to wide adoption. Simplicity is of course a great virtue, but does not guarantee correctness. The observant reader may already have spotted some warning signs: the gradient is only valid in the exact point it was computed. If we move a small epsilon away from that point, changes in gradient are possible. So perhaps if the neural network is sufficiently non-linear, the explanations of Grad-CAM cannot be trusted.

Already in literature doubts were raised about the correctness of several explanation methods [26–28]. For example, Ghorbani *et al.* [28] investigate the sensitivity of Grad-CAM's output with respect to input perturbations. They found that imperceptible perturbations of input images can falsify explanations, sharing some similarities with adversarial attacks on convolutional neural networks. We focus on a different question: is it possible to build neural networks whose Grad-CAM explanation is misleading while the network still performs well? With other words, does Grad-CAM provide robust explanations?

Figure 1.2.: A traffic sign that is misclassified by a deep network. The deep network contains a backdoor that causes it to misclassify the sign if a sticker is present (courtesy of [29]).

Why would we focus on such misleading or, what we call, lying networks that provide misleading explanations but do have good performance? It turns out that neural networks can be constructed with "backdoors" [29, 30]. Such networks seem to function normally, except when a particular pattern is present in the input, in that case its usual output is overruled and a false output is generated. An example is given in Figure 1.2, where a printed sticker on a stop sign will lead a network to misclassify the sign as a speed limit.

The danger is that because network weights are hard to interpret, it is not clear whether a backdoor is present in a neural network. Thus, such backdoored networks could be spread on the internet easily without detection. This is particularly concerning since so many machine learning models are shared and distributed through the internet.

It turns out that networks can in a similar way be backdoored such that their predictions remain correct, but their explanation provided by Grad-CAM are manipulated. We show this in Chapter 2. Such attacks are also hard to detect, because the accuracy of the network remains unchanged, and such manipulations can be made only to occur when particular input patterns are present. One can imagine that such a misleading networks could be dangerous if combined with Grad-CAM to locate objects such as pedestrians for self-driving cars.

1.6. ACTIVE LEARNING

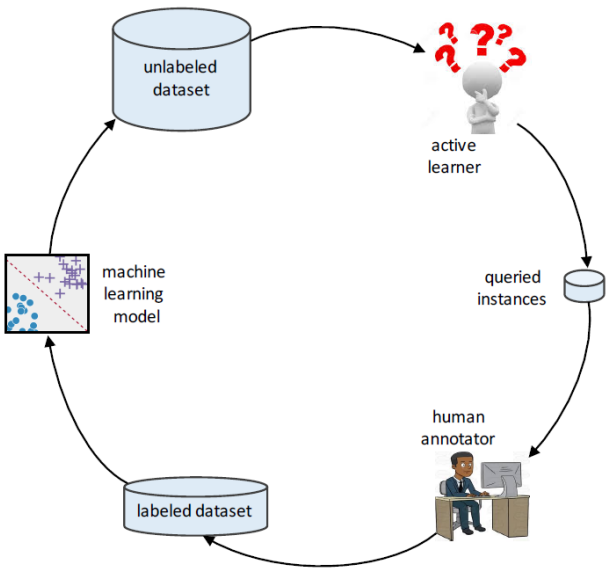


Figure 1.3.: The active learning procedure (courtesy of [31]).

The second topic we touch upon is active learning. In active learning, the learning procedure is iterative and proceeds in rounds and is illustrated in Figure 1.3. First, the active learner receives a, typically large, sample from $P_{\mathscr{X}}$, the marginal distribution of $P_{\mathscr{X}\mathscr{Y}}$. This sample is called the unlabeled pool, and it cannot yet be used for learning, as the outputs are not known. The active learning algorithm then chooses one or more samples from this pool, these are called the queries, after which an oracle provides the corresponding labels. Then a learning algorithm can build a model using the labeled sample to predict the corresponding y for unseen objects. Usually, there are multiple rounds where this is repeated: labels are requested, model is trained, labels are requested, etc. The process stops at some

point, when for example the labeling budget is exhausted or other stopping conditions are met. In an applied setting the labeling is done by a human annotator and thus only a finite amount of labels can be requested.

A natural candidate to compare active learners against is supervised learning, which in the context of active learning also referred to as random sampling. When sampling objects uniformly at random from the pool, the obtained training sample is i.i.d. again as in regular supervised learning. The goal is for the active learning strategy to select objects in a clever way to get a better performance with the same budget as compared to random sampling. Or the active learner may reach the desired performance sooner, reducing the labeling budget.

Active learning is particularly attractive if a lot of unlabeled data is available and if labels have a large cost. That cost can arise because domain experts are necessary to classify the object (e.g. doctors in a medical setting), but can also arise because annotation is time consuming or difficult. Sentiment classification is a practical example of a realistic setting. Here Twitter provides an extremely large unlabeled dataset, and services such as Mechanical Turk can be straightforwardly used to solicit labels.

That active learning can help and lead to improved performance compared to random sampling can be illustrated using artificial examples. For example, if we are performing classification in 1D with a linear separator, and the underlying true rule is also linear, active learners can find the decision boundary up to a precision epsilon with a budget that is logarithmically smaller than that of random sampling [32]. This is because the active learner can perform a binary search on the line to find the optimal threshold. However, in general active learners turn out to be hard to analyze, in part due to their biased or non i.i.d. selection of objects for labeling.

In the context of safety, it is natural to compare active learners to the baseline of supervised learning. We may wonder whether active learning is safe generally. Furthermore, can we say anything about when one active learning strategy beats another? This is the questions we set out to answer.

To be able to prove anything, we analyze an idealized setting where active learning strategies are used that minimize generalization bounds (one based on Rademacher complexity, but similar to the PAC one discussed in the preliminaries). In the theoretical machine learning community, the tightest generalization bounds are usually sought after (where the upperbound is as small as possible). This is because, in a worst-case scenario, the performance of the learning algorithm will reach the upperbound. Thus, if we have two algorithms, and we have to choose which to use, and we want to hedge against the worst-case scenario, we would choose the algorithm with the smallest upperbound. Does this then mean also that such an active learner is the safest to use in our notion of safety?

In Chapter 3, we analyzed three active learning algorithms, and we compared their corresponding learning bounds in terms of tightness. As motivated by the argument above, we could expect that the tightest bound would also lead to the best active learning performance. Perhaps surprisingly, we found the somewhat counter-intuitive result that active learners minimizing the loosest bounds performed the best. Thus tighter worst-case bounds do not imply that one algorithm will outperform the other. In particular, one should take into account whether such a worst-case event is likely to occur, and thus a tighter worst-case bound is not sufficient to imply safety. Using a an average-case analysis instead of a worst-case analysis we can explain our results.

1

1.7. LEARNING CURVES

While in the previous we focused on active learning, a setting which is generally already hard to analyze, in the last part we focus on the most fundamental form of safety of machine learning. The safety of supervised learning itself. We compare the performance of supervised learners with itself, trained on varying amounts of data.

In the machine learning community it seems to be that the conventional wisdom is: "the more data, the better"[2]. But is that really always the case? Is supervised learning safe, in the sense that, more data always leads to better models?

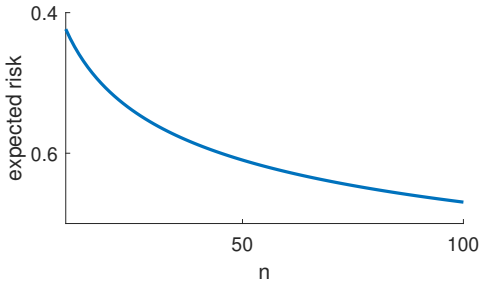


Figure 1.4.: Example of an idealized learning curve.

Learning curves are *the* tool to study such questions, an example of such a curve is given in Figure 1.4. Such curves plot the expected risk versus the sample size n. Learning curves are a useful tool for any machine learning practitioner. The curve can tell us many things about the learning problem at hand. For example, by extrapolating the curve we can see whether it is worth it to gather additional data. In particular, if dataset collection needs to be planned beforehand, this can be useful to estimate the minimum amount of data needed [33]. On the other hand, if our dataset is gigantic and this leads to all kinds of computational issues, the learning curve can inform us whether subsampling the dataset can be used to speed up computations [34].

The question then, if more data is always better, can then also be framed in the context of the monotonicity of the learning curve. Since lower risk is better, the question becomes, is the curve (strictly) monotonically decreasing? Perhaps surprisingly, the answer turns out to be no in several cases. Thus, in a sense, machine learning algorithms are in general not safe: more data could be dangerous and deteriorate their performance. Or in other words, the curve can display non-monotonicity or is, what we call, ill-behaved. For example, see the curve in Figure 1.5 which suffers from a phenomenon that is referred to as peaking.

The three chapters on the topic of learning curves are largely devoted to (non)-monotonicity. Examples of such ill-behavior have been known for already quite some time and part of this thesis (Chapter 4) aims to collect all known results regarding such phenomena. We survey the literature, describe and categorize findings related to the shapes of learning curves. It turns out that there is some empirical and theoretical evidence that favors learning curves to have a power law or exponential shape. We call such curves well behaved, indicating that they are monotonically improving with more data. On the other hand, we also found

[2]We give several citations in Section 4.6 of Chapter 4 to further support this claim.

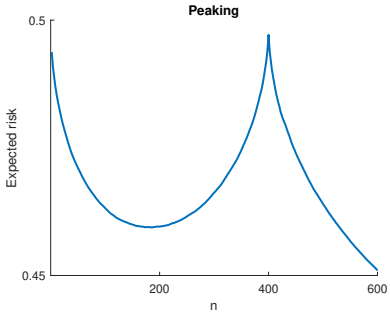


Figure 1.5.: Example of an ill-behaved learning curve. This example of peaking has already been known since 1989 [12].

strong evidence that shows learning curves can have various other shapes that are much harder to characterize, such as learning curves with multiple peaks. We put these shapes in 6 categories and discuss their (potential) causes and how to possibly mitigate them. The Bayesian variant of the learning curve, that we call a PA learning curve, is sometimes easier to analyze analytically, and as such there are various studies that try to characterize their shapes. We find that only for a very limited class of problems and algorithms there are proofs that guarantee monotonicity. Meanwhile, it is still relatively uncertain how widespread non-monotonicity is in real-world settings. Preliminary results by Mohr *et al.* [35] indicate that non-monotonicity occurs for some specific learners and may happen less often for large sample sizes. However, we believe a deeper investigation is necessary.

Besides the fact that ill-behaving learning curves are a curious phenomenon because they are so unexpected, generally speaking, data is often valuable in the sense that it can be difficult, expensive, or time-consuming to collect. Most annotated data does not come for free and a significant amount of resources are spent nowadays on collecting data. In that light, unsafe learners that potentially deteriorate in performance with more data are quite undesirable. Ideally, we would use learners that always make good use of additionally collected data and continuously improve their performance. Besides the practical concern, it seems conceptually appealing to have learners that always improve with more data. Besides that, the question is of theoretical interest since it is not at all clear whether this is possible generally.

We set out initially to find more examples of monotone learners in simple settings. We mainly looked at simple settings to keep the mathematical analysis tractable, for example, regression and classification in 1D. However, even proving monotonicity for such cases turned out to be very difficult and we got stuck on this problem for long periods of time. At some point we turned it around, and tried to find examples that are non-monotone. And indeed, it turns out there are many more novel simple problems that are non-monotone as well, as we find in Chapter 5. Even for simple settings, we managed to find very remarkable learning curves, see Figure 1.6. This learning curve shows periodicity, where the risk worsens and recovers multiple times.

So, do we only offer more negative examples regarding monotonicity? No, there is also reason to be optimistic. Learning curves are analyzed by supplying the learner with more

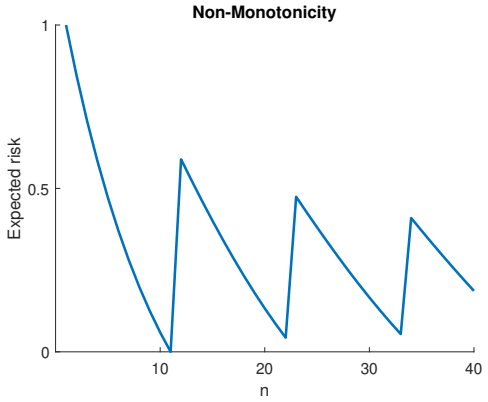


Figure 1.6.: Example of a remarkable learning curve from Chapter 5.

and more data and measuring its performance on the true distribution. But maybe such a theoretical analysis is too idealized to accurately characterize how machine learning works in practice...

In practice, when we are faced with a stream of incoming data, we would usually not keep iterating the learning procedure to build models on more and more training data and directly apply them. Instead, a careful practitioner will continually estimate the performance of the various models they train, for example using cross validation or the holdout method. Generally, a careful practitioner would probably notice that a newly deployed model would show significantly worse performance, and would resort to the previously best trained model. Such evaluations and withholding of worse models while constructing the learning curve, can avoid that the (non-averaged) learning curve increases significantly, and one may expect that the learning curve becomes more well-behaved or more monotone, meaning that the risk increases less often or in smaller amounts.

We investigate this idea in more depth and formally analyze it in Chapter 6. We propose a wrapper algorithm that can be applied on top of any supervised learner which aims to make the learning curve more monotone. In this setting there is a streaming source of data that is incoming to the wrapper algorithm. The wrapper algorithm sets aside some of the incoming data as test data and uses that to evaluate if newly trained models perform significantly better. To compare models, a hypothesis test is used, which allows us to derive theoretical guarantees regarding monotonicity of the learning curve generated by this procedure. As such, this wrapper algorithm can be seen as a first step towards making any learning algorithm more monotone and thus more safe.

1.8. RELATION OF THE CHAPTERS TO SAFETY

While not explicit, all chapters are tied together by the concepts of safety. In the context of explainability, we study whether explanations of Grad-CAM are robust, in the sense of whether they can be manipulated. For active learning, we study safety guarantees given by worst-case versus average-case approaches. Finally, we return to the core of machine learning with the topic of learning curves. Here we study the safety of supervised learning compared with itself for varying sample sizes. Here we review all kinds of examples and find new ones where supervised learning itself is not safe, or in other words, where the learning curve is non-monotone. Finally, we make a constructive step towards making supervised learning safe by building a wrapper algorithm which aims to make the learning curve monotone.

1.9. ORGANIZATION OF THE THESIS

First, we start off with Chapter 2 regarding the safety of the explanation method Grad-CAM for convolutional neural networks. In Chapter 3 we analyze active learners based on generalization bounds.

Subsequently, we move on to the topic of learning curves. In Chapter 4 we give a survey of the literature regarding learning curves, their basics such as definition and estimation, and empirical and theoretical works regarding their shapes and monotonicity. This includes several examples of non-monotone learning curves. Then, Chapter 5 adds to that, by providing several examples of novel and simple problems that have non-monotone learning curves. We wrap up the topic of learning curves with Chapter 6 which introduces a wrapper algorithm which aims to make the learning curve more monotone for any learner. We conclude the dissertation with Chapter 7 which offers an extensive discussion and directions for future work.

Finally, this thesis also offers several appendices. A precursor of Chapter 5 was an extended abstract which can be found in Appendix A. Besides the survey on learning curves, we also worked to trace back the historical origins of the peaking phenomena of learning curves (of Figure 1.5) which recently found renewed interest in the context of Double Descent [36]. This lead to the letter [12] which is given in Appendix B. Appendices C and D contain the proofs and more details of Chapters 3 and 5, respectively. In Appendix E we present an additional theoretical result related to Chapter 7, where we compute the exact learning curve distribution for a wrapper algorithm.

1.10. OVERVIEW OF NOTATION

We have aimed to use as much similar notation between the chapters. Only Chapter 2 uses different notation, since this chapter considers Grad-CAM which is applied to convolutional neural networks, meanwhile the rest of the dissertation focuses more on machine learning. The tables on the next pages may prove useful when it comes to notation. Finally, we provide the list with the most important terminology that can be encountered in this dissertation (glossary).

1

Table 1.1.: Notation used in this dissertation

Domain	
$\mathcal{X}$	The domain of the input features (often $\mathbb{R}^d$, where d is the dimensionality)
$\mathcal{Y}$	The domain of the outputs (classification: $\mathcal{Y} \in \{-1, +1\}$, regression $\mathcal{Y} \subset \mathbb{R}$)
$\mathcal{Z}$	Combined domain, $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$
$\mathcal{S}$	All possible training sets, $\mathcal{S} := \mathcal{Z} \cup \mathcal{Z}^2 \cup \mathcal{Z}^3 \cup \ldots$
Problem	
f	Underlying true labeling function $f : \mathcal{X} \to \mathcal{Y}$
P_{XY}	Distribution over $\mathcal{X} \times \mathcal{Y}$ (often shortened to P)
P_X	Marginal density over $\mathcal{X}$ (obtained by marginalizing out the Y in P_{XY})
$\mathcal{P}$	Distribution over P_{XY} (distribution over distributions)
Data	
x, x_i, x'	Feature vector $\in \mathcal{X}$
y, y_i, y'	Label $\in \mathcal{Y}$
$\hat{y}$	Prediction of the label for some object
S_n	Training set of (x, y) pairs of size n
S^k_{n-k}	A specific training set of size n
n	Sample size of the training set
$\widehat{P}$	Sample from P_X (unlabeled sample)
$\widehat{Q}, \widehat{Q}_n$	A sample selected from $\widehat{P}$ by the active learner (of size n)
$\widehat{Q}^{\text{lab}}_n$,$\widehat{P}^{\text{lab}}$	The corresponding labeled sample (e.g. (x, y) pairs)
$n_{\widehat{P}}, n_{\widehat{Q}}$	Size of the sample $\widehat{P}$, $\widehat{Q}$
$X_{\widehat{P}}$	$n_{\widehat{P}} \times d$ datamatrix of features of $\widehat{P}$
$X_{\widehat{Q}}$	$n_{\widehat{Q}} \times d$ datamatrix of features of $\widehat{Q}$
X, X_n	$n \times d$ datamatrix of features
Y	n-dimensional vector of labels
Learner	
K, $K(x, x')$, $k(x, x')$	a positive semi-definite (PSD) kernel of the learner
$K(X_n, X_n)$	an $n \times n$ kernel matrix corresponding to the features of X_n
$K_{\mathcal{L}}$, $K_{\mathcal{L}}(x, x')$	a positive semi-definite (PSD) kernel for the MMD
$\mathcal{H}$	the reproducing kernel Hilbert space (RKHS) of K
$\|\|.\|\|_K$	Norm of object in $\mathcal{H}$
H	A hypothesis set (all models that the learner can return) example: $H = \{x \to w^T x + b : w \in \mathbb{R}^d, b \in \mathbb{R}\}$
w	parameter of linear model
A	Learner that maps a training set to a model, $A : \mathcal{S} \to H$
A_{erm}, A_{reg}	Empirical Risk Minimizer (ERM), Regularized Loss Minimizer
h, h_i, h_{best}	models from H
μ	regularization parameter

Table 1.2.: Notation used in this dissertation (continued)

Performance	
l	a loss function
l_{01}	zero-one loss
l_{sq}	squared loss
$\epsilon(h)$	the true error (error rate) of h, $\mathbb{E}_{(x,y)\sim P} l_{01}(h(x), y)$
$L_{\widehat{Q}}(h, f)$,$L_{\widehat{P}}(h, f)$	loss of h on $\widehat{Q}, \widehat{P}$ with label provided by f
$\widehat{\epsilon}(h, S)$	empirical error (average 01 loss) of h on S
$R(h)$	risk of h, $\mathbb{E}_{(x,y)\sim P} l(h(x), y)$
$\bar{R}_n(A)$	expected risk of learner A: $\mathbb{E}_{S_n \sim P^n} R(A(S_n))$
$\bar{R}_n^{\text{PA}}(A)$	problem-average risk, e.g. $\mathbb{E}_{P\sim\mathscr{P}} \bar{R}_n(A)$
PAC Learning and bounds	
$R_m(H)$	Rademacher complexity of H
δ	probability that the bound does not hold
Miscellaneous	
λ, λ_i	eigenvalue ordered by absolute size $\lvert\lambda_1\rvert \geq \lvert\lambda_2\rvert \ldots$
α	confidence level of a hypothesis test
H_0, H_1	null hypothesis and alternative hypothesis
η	measures the size of the approximation error made by H
σ	can also refer to the lengthscale of the RBF kernel see Chapter 3
Λ	upperbound on the norm of a model $h \in H$ see Chapter 3
$\lvert\lvert.\rvert\rvert_\infty, \lvert\lvert.\rvert\rvert_2, \lvert\lvert.\rvert\rvert_1$	infinity/maximum norm, euclidean norm, taxicab norm

1

GLOSSARY

1NN,kNN

1-nearest neighbour, k-nearest neighbour (classifier).

A high probability result

A result that can be made to hold with arbitrarily high probability (by increasing the sample size).

Agnostic

The opposite of realizeable; thus the approximation error will be non-zero.

AULC

Area Under the Learning Curve.

Batch Active Learning

The setting in active learning where samples are labeled in batches, e.g. 5 samples at a time.

Bayes decision boundary

The boundary found by the Bayes classifier.

Bayes error

The theoretical optimal error of Bayes classifier.

Bayes optimal model or Bayes classifier

The optimal model in theory (based on knowledge of the unknown distribution).

Bayesian

Someone who adheres to the Bayesian statistical school of thought.

C4.5

An algorithm to grow a Decision Tree.

CNN

Convolutional Neural Network.

Conjugate prior

A conjugate prior is a specific choice of prior that depends on the likelihood function. If a conjugate prior is used, the posterior can be computed analytically.

Cross Validation

An algorithm that splits datasets into folds of training sets and test sets, which is typically used to tune hyperparameters or estimate the generalization error or risk.

Curse of Dimensionality

See Peak Effect.

Discrepancy

Similarity measures that measure similarities between 2 empirical samples in the space $\mathscr{X}$. Similar to the Maximum Mean Discrepancy (MMD) and Nuclear Discrepancy (ND).

Distribution

If no further indication is given, it means the distribution P_{XY} that generates the datasets.

Double Descent

The local maximum in the feature curve (or complexity curve) of a machine learning model. For more detail, see Section 4.2.7.

ERM

Empirical Risk Minimizer, a learner minimizes the empirical risk on the training data.

Excess Risk

The risk of a model h compared to the best model on the underlying distribution. The first term in Equation 1.3.

Featuremap

An intermediate presentation of a CNN that has the a tensor shape. The spatial locations in the featuremap can be mapped back to spatial locations in the input image. Typically, a featuremap has many more channels than the input image.

Frequentist

Someone who adheres to the Frequentist statistical school of thought.

Generalization Bound

Similar to a PAC result, but generalization bounds are slightly broader defined (e.g. they can also be used in other settings than the PAC setting).

Generalization Error

Synonym for Risk.

Gibbs

A Bayesian learner that samples models from the posterior distribution (it is randomized).

GradCAM

An algorithm which tries to assign each pixel in an image a value such that it is indicative of that pixels' influence in a CNN when it comes to its prediction.

Hinge loss

The loss typically used to train an SVM (support-vector-machine, a popular classification model).

Hold Out Method

The method where the data is split into two parts, a training set and test set, to estimate the risk or to tune hyperparameters.

Hughes Phenomenon

See Peak Effect.

Hypothesis class

The set of all models that could be returned by the learner.

i.i.d.

Independent and identically distributed. Meaning that each variable has the same distribution and is independently sampled.

Ill-behaved

Synonym for a non-monotone learning curve. Meaning performance may detoriate in expectation when increasing the training set size.

KLSC

Kernelized Least Squares Classifier. A classifier obtained by minimizing the squared loss on the training set and a L_2 regularization term.

Learning Rate

The rate with which the upperbound on the error decreases (given by generalization bound) if we get additional samples. For example, in the realizeable setting, we have that the error decreases as $O(1/n)$, in the agnostic case we have $O(1/\sqrt{n})$ (based on a PAC analysis).

Marginal distribution

The marginal distribution P_X. This distribution is obtained by marginalizing out the Y from P_{XY} and models the distribution of unlabeled data.

Miss-specified

The data generation procedure assumed by the probabilistic model is not correct.

MNLR

The same model as PFLD, but MNLR stands for minimum norm linear regression.

Model misspecification

The data generation procedure assumed by the probabilistic model is not correct, or we are in the agnostic case.

Monotonicity

We call a learning curve monotone if the curve improves its performance in expectation when increasing the training set size.

MSE

Mean squared error (the average squared loss on a dataset).

PA

Problem-Average. There is an average over multiple problems to come to the risk or learning curve (meaning a distribution over distributions). See also $\mathscr{P}$ in the notation table or Equation 1.11.

PAC

Probably Approximately Correct. See Section 1.2 (page 3), for more detail please refer to [9, Chapter 3].

Peak Effect

The classical expected U-shape of a feature-curve as predicted by the bias-variance trade-off.

Peaking

The local maximum in the learning curve of a machine learning model. This was first discovered for Pseudo-Fisher (see PFLD) but also seems to occur for deep neural networks. For more detail, see Section 4.6.2.

Peaking of Feature Curves

See Peak Effect.

PFLD

Pseudo-Fisher Linear Discriminant, or Pseudo-Fisher. A model that minimizes the squared loss on the training set for classification. If $d > n$, it uses the Pseudo-Inverse to find a unique solution (the minimum norm solution).

Rademacher Complexity

A measure of complexity for a hypothesis class similar to the VC dimension.

Realizeable

Meaning that the hypothesis class H is rich enough that the true (unknown) model is part of it. In this case, the approximation error will be zero.

Risk

The performance on the underlying distribution $P_{\mathscr{X}\mathscr{Y}}$ of the learning problem. Typically, the risk needs to be estimated using unseen test data in a practical setting, since in that case the distribution is not known.

RKHS

Reproducing Kernel Hilbert Space. This is the high-dimensional space that corresponds to a PSD (positive semi-definite) kernel.

Sample Complexity

The amount of samples necessary for successful learning, meaning the amount of necessary to achieve a particular excess risk. Technically, $m(\epsilon,\delta)$ in the PAC definition (see for instance [9, definition 3.1]).

Single-Shot

That all samples for labeling in an active learning setting are selected in a single time. So there is no interaction between labeler and active learning algorithm.

Structural Risk Minimization

An algorithm that considers multiple hypothesis classes. It selects a hypothesis class such that the models provide a good fit on the training set, while avoiding hypothesis classes that are too complex and lead to overfitting. Thus, SRM makes a trade-off between approximation and estimation error.

Surrogate loss

Because the l_{01} loss is tough to optimize, we often resort to a different loss which is more easily optimized (e.g. squared loss).

VC Dimension

A measure of complexity for a hypothesis class.

Well-behaved

Synonym for monotonicity. We call a learning curve monotone if the curve improves its performance in expectation when increasing the training set size.

Well-specified

Meaning that the data generation procedure assumed by the probabilistic model (i.e. the likelihood, prior) are correct.

Wrapper Algorithm

An algorithm that, as part of its routines, calls another learning algorithm. The wrapper algorithms studied in this dissertation aim to make the learning curve of the algorithm which is being wrapper more monotone.

1.11. BIBLIOGRAPHY

[1] K. Johnson, *Neurips requires ai researchers to account for societal impact and financial conflicts of interest.* VentureBeat (2020).

[2] H. Kuwajima, H. Yasuoka, and T. Nakae, *Engineering problems in machine learning systems,* Machine Learning , 1 (2020).

[3] *Safe machine learning workshop at iclr 2019,* `https://sites.google.com/view/safeml-iclr2019/` ().

[4] *Safe machine learning workshop at ecai 2020,* `https://safeml.bitbucket.io/` ().

[5] P. A. Ortega, V. Maini, and D. S. Team, *Building safe artificial intelligence: specification, robustness, and assurance,* DeepMind Safety Research Blog (2018).

[6] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, *Concrete problems in ai safety,* arXiv preprint arXiv:1606.06565 (2016).

[7] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, *Evasion attacks against machine learning at test time,* in *Joint European conference on machine learning and knowledge discovery in databases* (Springer, 2013) pp. 387–402.

[8] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, *Adversarial attacks and defences: A survey,* arXiv preprint arXiv:1810.00069 (2018).

[9] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms* (Cambridge university press, 2014).

[10] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, Vol. 4 (Springer, 2006).

[11] R. P. Duin, *Classifiers in almost empty spaces,* in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, Vol. 2 (IEEE, 2000) pp. 1–7.

[12] M. Loog, T. Viering, A. Mey, J. H. Krijthe, and D. M. Tax, *A brief prehistory of double descent,* Proceedings of the National Academy of Sciences **117**, 10625 (2020).

[13] K. P. Murphy, *Machine learning: a probabilistic perspective* (MIT press, 2012).

[14] G. J. McLachlan, *Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis,* Journal of the American Statistical Association **70**, 365 (1975).

[15] J. H. Krijthe and M. Loog, *Optimistic semi-supervised least squares classification,* in *2016 23rd International Conference on Pattern Recognition (ICPR)* (IEEE, 2016) pp. 1677–1682.

[16] S. B. David, T. Lu, T. Luu, and D. Pál, *Impossibility theorems for domain adaptation,* in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (JMLR Workshop and Conference Proceedings, 2010) pp. 129–136.

[17] M. Loog, *Contrastive pessimistic likelihood estimation for semi-supervised classification,* IEEE transactions on pattern analysis and machine intelligence **38**, 462 (2015).

[18] J. H. Krijthe and M. Loog, *Projected estimators for robust semi-supervised classification,* Machine Learning **106**, 993 (2017).

[19] W. M. Kouw and M. Loog, *Target contrastive pessimistic discriminant analysis,* arXiv preprint arXiv:1806.09463 (2018).

[20] J. Krijthe and M. Loog, *The pessimistic limits and possibilities of margin-based losses in semi-supervised learning,* in *Advances in Neural Information Processing Systems* (2018) pp. 1790–1799.

[21] A. Adadi and M. Berrada, *Peeking inside the black-box: A survey on explainable artificial intelligence (xai),* IEEE Access **6**, 52138 (2018).

[22] V. Polonski, *Ai is convicting criminals and determining jail time, but is it fair,* in *World Economic Forum* (2018).

[23] A. Selbst and J. Powles, *meaningful information and the right to explanation,* in *Conference on Fairness, Accountability and Transparency* (PMLR, 2018) pp. 48–48.

[24] R. C. Fong and A. Vedaldi, *Interpretable explanations of black boxes by meaningful perturbation,* in *Proceedings of the IEEE International Conference on Computer Vision* (2017) pp. 3429–3437.

[25] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, *Grad-cam: Visual explanations from deep networks via gradient-based localization,* in *Proceedings of the IEEE international conference on computer vision* (2017) pp. 618–626.

[26] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, *Sanity checks for saliency maps,* in *Advances in Neural Information Processing Systems* (2018) pp. 9505–9515.

[27] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim, *The (un) reliability of saliency methods,* in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (Springer, 2019) pp. 267–280.

[28] A. Ghorbani, A. Abid, and J. Zou, *Interpretation of neural networks is fragile,* in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33 (2019) pp. 3681–3688.

[29] T. Gu, B. Dolan-Gavitt, and S. Garg, *Badnets: Identifying vulnerabilities in the machine learning model supply chain,* arXiv preprint arXiv:1708.06733 (2017).

[30] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, *Trojaning attack on neural networks,* (2017).

[31] Y. Yang, *Towards Practical Active Learning for Classification*, Ph.D. thesis, Delft University of Technology (2018).

[32] B. Settles, *Active learning literature survey*, Tech. Rep. (University of Wisconsin-Madison Department of Computer Sciences, 2009).

[33] L. J. Frey and D. H. Fisher, *Modeling decision tree performance with the power law.* in *AISTATS* (1999).

[34] F. Provost, D. Jensen, and T. Oates, *Efficient progressive sampling,* in *ACM SIGKDD* (1999) pp. 23–32.

[35] F. Mohr, T. J. Viering, M. Loog, and J. N. van Rijn, *LCDB 1.0: An extensive learning curves database for classification tasks,* in *Machine Learning and Knowledge Discovery in Databases, ECMLPKDD*, Lecture Notes in Computer Science (Springer, 2022) p. accepted.

[36] M. Belkin, D. Hsu, S. Ma, and S. Mandal, *Reconciling modern machine-learning practice and the classical bias–variance trade-off,* Proceedings of the National Academy of Sciences **116**, 15849 (2019).

2

How to Manipulate CNNs to Make Them Lie: the GradCAM Case

Recently many methods have been introduced to explain CNN decisions. However, it has been shown that some methods can be sensitive to manipulation of the input. We continue this line of work and investigate the explanation method GradCAM. Instead of manipulating the input, we consider an adversary that manipulates the model itself to attack the explanation. By changing weights and architecture, we demonstrate that it is possible to generate any desired explanation, while leaving the model's accuracy essentially unchanged. This illustrates that GradCAM cannot explain the decision of every CNN and provides a proof of concept showing that it is possible to obfuscate the inner workings of a CNN. Finally, we combine input and model manipulation. To this end we put a backdoor in the network: the explanation is correct unless there is a specific pattern present in the input, which triggers a malicious explanation. Our work raises new security concerns, especially in settings where explanations of models may be used to make decisions, such as in the medical domain.

This work was accepted at the BMVC 2019: Workshop on Interpretable and Explainable Machine Vision, Cardiff, UK [1].

2.1. Introduction

For deep convolutional neural networks, it is difficult to explain how models make certain predictions. Explanations for decisions of such complex models are desirable [2]. For example, in job application matching, explanations may reveal undesireable biases in machine learning models. For settings which demand rigorous security demands such as self driving cars, explanations can help us better understand how models work in order to identify and fix vulnerabilities. In other application domains, such as neuroscience, machine learning is not only used for predictions (e.g., regarding a disease), but also to understand the cause (the underlying biological mechanism). Explanations can thus help experts discover new phenomena.

The field of Explainable AI (XAI) aims to tackle this problem; how did a particular model come to its prediction? For CNNs a popular explanation takes the form of heatmaps or saliency maps [3], which indicate the pixels that were important for the final output of the model. Recently, many explanation techniques have been proposed in the literature to generate explanations for machine learning models [3–19]. A nice introduction and survey to the XAI is [2].

Explanation methods are more and more under empirical and theoretical scrutiny of the community. For example, Ancona *et al.* [20] show equivalence and connections between several explanation methods, and Lundberg and Lee [4] unify six existing explanation methods. Several studies [6, 7, 21–23] have raised questions regarding robustness and faithfulness of these explanations methods. For example, Ghorbani *et al.* [23] show that an adverserial imperceptible perturbations of the input can change the explanation significantly while the model's prediction is unchanged.

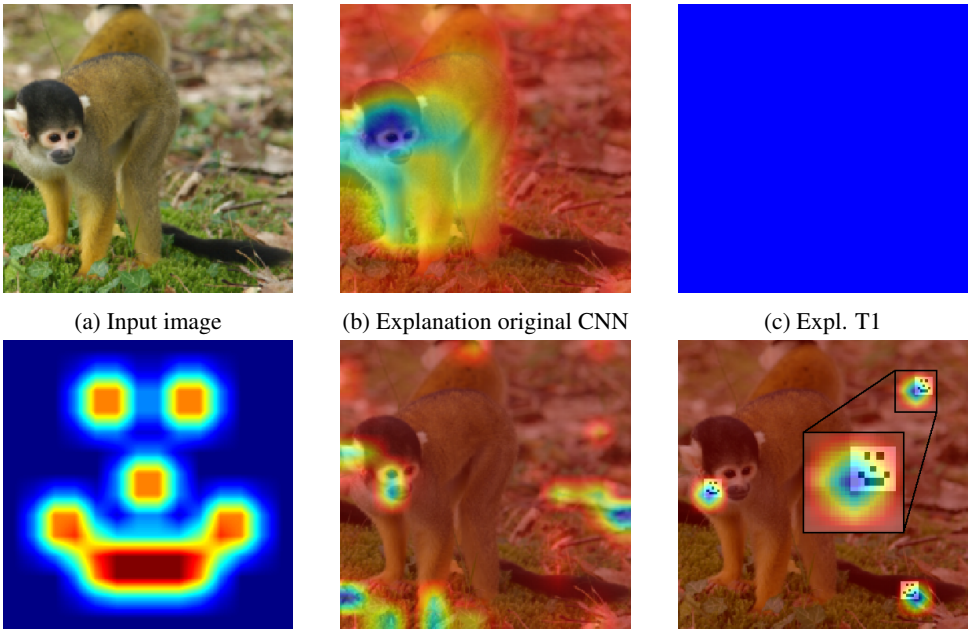
We continue this line of investigation and uncover new (security) vulnerabilities in the popular explanation method GradCAM [5]. GradCAM, a generalization of the explanation method CAM [24], is a fast and simple method to explain CNN decisions and is applicable to many CNN architectures. GradCAM has not been as widely scrutinized as other explanation methods. Adebayo *et al.* [21] propose several sanity checks that should be satisfied by explanation methods, e.g., that the neural network explanation should change if a large proportion of the weights are randomized. Adebayo *et al.* [21] find GradCAM satisfies their proposed checks, motivating further study of this explanation method.

Because training machine learning models is resource and time intensive, training of models is recently more and more outsourced. It is now possible to upload training data and model architecture, and to train the model in the cloud, for example using platforms created by Google [25], Amazon [26] or Microsoft [27]. It is expected that this will become the norm. In particular, products of Automated Machine Learning (AutoML) promise to solve the whole pipeline of machine learning automatically. The user only has to upload the dataset, and the cloud provider will automatically try several architectures, tune hyperparameters, train models, and evaluate them [28]. Another approach to circumvent costly training procedures is to finetune existing models for new tasks [29].

Both outsourcing and finetuning pose a security risk [30]. Gu *et al.* [30] show in their case study with traffic signs, that by manipulating the training data, the model will misclassify stop signs if a sticker is applied to them. Liu *et al.* [31] introduce a technique that can be applied to an already trained model to introduce malicious behaviour. Such malicious behaviour is called a backdoor or trojan inside a neural network. The backdoor is triggered

by specific input patterns while keeping model performance on the original task more or less the same. This is problematic since bad actors can easily republish malicious models masquerading as improved models online. Because of the blackbox nature of deep learning models, such trojans are difficult to detect [32, 33]. Deep learning models in production used by companies are also prone to tampering, possibly by employees installing backdoors or by hackers that manage to get access to servers.

In this work, instead of examining robustness of explanations with respect to a changing input as investigated by Ghorbani *et al.* [23], we investigate the robustness of explanations when the model is modified by an adversary such as the scenario considered by Liu *et al.* [31] and Wang *et al.* [32]. Our work can be considered as a white-box attack on the explanation method GradCAM and the model [34].



(a) Input image (b) Explanation original CNN (c) Expl. T1

(d) Expl. T2 (e) Expl. T3 (f) Expl. T4

Figure 2.1.: Qualitative example of manipulated explanations for manipulated networks T1-T4. Blue means a pixel had a large influence on the decision. (c,d) The networks T1 and T2 generate always the same explanation, irrespective of the input to the network. (e) T3 generates a semi-random explanation based on the input. (f) T4 only generates a malicious explanation if a specific pattern (in this case, a smiley) is visible in the input. The area in the square for is enlarged.

Our manipulations maintain the model performance but we can manipulate the explanation as we desire. An overview of our proposed techniques T1-T4 are shown in Figure 2.1. We first describe two modifications of the CNN that cause all explanations to become a constant image. Arguably, this manipulation is easy to detect by inspecting the explanations, which is not as easy for the two more techniques that we propose. In one of our techniques

the explanation is semi-random and depends on the input. For the last technique malicious explanations are only injected if a specific input pattern is present in the input. These last two techniques are much more difficult to detect using visual inspection of explanations and therefore pose a more serious security concern.

Several works use explanations to localize objects in images [3, 5, 15], which could be used by secondary systems; for example, as a pedestrian detector for a self-driving car or an explanations used by a doctor to find a tumor. Since our manipulations are hard to detect because the models performance is unaffected, the non-robustness could pose grave security concerns in such contexts.

Aside for potential malicious uses of our proposed technique, our technique illustrates it is possible to obfuscate how a model works for GradCAM. Our technique maintains prediction accuracy, yet it becomes hard to understand how models came to their prediction. Thus the model becomes impossible to interpret, while staying useful. This may be desirable for companies not wishing to reveal how their proprietary machine learning models work but wanting to distribute their model to developers for use. Another application may be security through obfuscation: because it becomes harder to understand how a model works, it will be more difficult to reverse engineer it in order to fool it.

2.2. GradCAM and Notation

We briefly review the notation and the GradCAM method [5]. We only consider CNNs for classification tasks. Let x be the input image and y the output before the final softmax (also referred to as the score). Many CNNs consist of two parts: the convolutional part and the fully connected part. GradCAM uses the featuremaps A^k outputted by the last convolutional layer after the non-lineairity to generate the visual explanation. Here $k = 1, \ldots, K$ indicates the channel, and a single A^k can be regarded as a 2D image. The visual explanation or heatmap I^c for a class c is computed by

$$I^c = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right). \tag{2.1}$$

Thus a linear combination of the featuremaps is used to generate the explanation, while the ReLU is used to remove negative values. α_k^c is obtained by global-average-pooling the gradient for class c with respect to the kth featuremap,

$$\alpha_k^c = \frac{1}{N_A} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k}, \tag{2.2}$$

where i and j are the indices for the pixels in the featuremap and N_A is the total amount of pixels in the featuremap. Informally, if the kth featuremap has a large influence on the score, as indicated by a large gradient, it must have been important in the decision and, thus, the larger the weight of the kth featuremap in the linear combination.

2.3. Manipulating the CNN

We will show several techniques that manipulate the architecture and weights to change the explanation of GradCAM, while keeping the performance of the CNN (more or less)

unchanged. The recipe for all these approaches will be the same. Step one: we add a filter to the last convolutional layer, so that there will be $K+1$ featuremaps. The $(K+1)$th featuremap will contain our desired target explanation I_T. We will scale A^{K+1} in such a way that $A^{K+1}_{ij} \gg A^k_{ij}$ for all pixel locations i, j and channels k. Step two: we change the architecture or weights of the fully connected part, to ensure $\alpha^c_{K+1} \gg \alpha^c_k$ for all c and k. Under these conditions, following Equation 2.1 and 2.2, the GradCAM explanation will be more or less equal to our desired target explanation, $I^c \approx I_T$ for all c. Figure 2.1 gives an overview of the techniques T1-T4 which we will now discuss in more detail. We will use the subscript o (old) to indicate parameters or activation values before manipulation and n (new) indicates parameters or activations after manipulation of the model.

2.3.1. TECHNIQUE 1: CONSTANT FLAT EXPLANATION

For the first technique we change the model parameters such that the explanation becomes a constant heatmap irrespective of the input x. Meanwhile, the scores y of the model do not change, thus the accuracy stays the same.

We manipulate the network as follows. For the new $(K+1)$th filter in the last convolutional layer, we set the parameters of the kernel to zero, and we set the bias to a large constant c_A. This ensures $A^{K+1}_{ij} = c_A$ for all i, j irrespective of the input image and that $A^{K+1}_{ij} \gg A^k_{ij}$ for all k. Let Z be the last featuremap in the convolutional part of the model. Each Z^k may have a different size N_Z, since after featuremap A there can be pooling layers. We assume there are only max / average pooling layers between A and Z, in that case $Z^{K+1}_{ij} = c_A$. Let z be the vector obtained by flattening the last featuremaps Z^k. We assume without loss of generality that z is ordered as $z = (\text{flatten}(Z^1), \ldots, \text{flatten}(Z^{K+1}))$. Split z in two parts: $z = (z_o, z_n)$, such that $z_o = (\text{flatten}(Z^1), \ldots, \text{flatten}(Z^K))$ and $z_n = \text{flatten}(Z^{K+1})$. Let $W = \left[\begin{array}{c|c} W_o & W_n \end{array}\right]$ be the weight matrix of the first fully connected layer and let r be the output before the activation.

$$r_o = W_o z_o + b_o,$$

where b_o is the old learnt bias. For the manipulated model

$$r_n = W_o z_o + W_n z_n + b_n.$$

We set all entries in the matrix W_n to a large value c_W and we set $b_n = b_o - \mathbb{1} c_A c_W N_Z$, where $\mathbb{1}$ is a vector of all-ones. Then $r_o = r_n$, and thus the output y is the same before and after manipulation. Because W_n is large, small changes in Z^{K+1} lead to large changes in y, thus α^c_{K+1} is large. This ensures $\alpha^c_{K+1} \gg \alpha^c_k$. Recall that however, Z^{K+1} is constant.

2.3.2. TECHNIQUE 2: CONSTANT IMAGE EXPLANATION

In the last technique, the target explanation I_T was a constant. Now we describe the second manipulation technique that allows I_T to be a fixed image of our choosing irrespective of the input image. We use the same technique as before, with two differences. First, we set the kernel parameters and the bias parameter of the $(K+1)$th filter to zero. Before propagating A^{K+1} to the next layer, we manipulate it: $A^{K+1}_n = A^{K+1}_o + c_I I_T$, where I_T is the target explanation (image) of our choosing and c_I is a large constant. This can be

seen as a architectural change. We set all values in W_n to a large value c_W and we set $b_n = b_o - \mathbb{1} c_W S_Z$, where $S_Z = \sum_{ij} Z_{ij}^{K+1}$ (note S_Z is independent of x). Then again $r_o = r_n$, and thus $y_o = y_n$. The arguments of the previous technique still hold and thus we have $A_{ij}^{K+1} \gg A_{ij}^{k}$ and $\alpha_{K+1}^{c} \gg \alpha_{k}^{c}$.

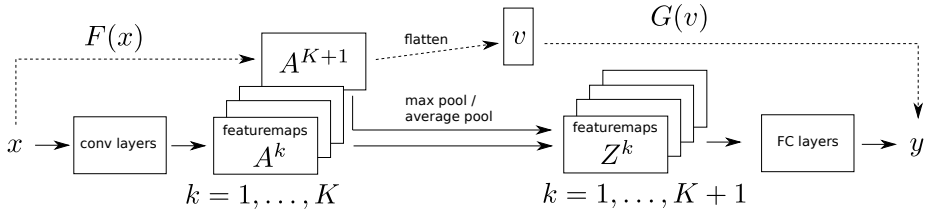


Figure 2.2.: Illustration of architectural changes necessary for techniques T3 and T4. Dashed lines indicate modifications. 'conv layers' indicates the convolutional part of the CNN, and the 'FC layers' indicate the fully-connected part of the CNN.

2.3.3. Technique 3: Semi-Random Explanation

A limitation of the previous techniques is that the explanation is always the same irrespective of the input. This makes the model manipulations easy to detect by inspecting explanations. Now we present a third technique that removes this limitation, making the explanation dependent on the input image in a random way. Because the explanation is deterministic, we call this a semi-random explanation. Making the explanation dependent on the input however comes with a price: the scores y may change a small amount of ϵ and more architectural changes to the model are required. The architectural changes are illustrated in Figure 2.2.

As before we will put our target explanation I_T in A^{K+1}. Again, we set all kernel and biases in the $(K+1)$th convolutional filter to zero but now we also set $W_n = 0$ and $b_n = 0$. To put the target explanation in A^{K+1}, we set $A_o^{K+1} = A_n^{K+1} + c_F F(x)$, where $F(x)$ will be a neural network taking x as input and outputs our desired target explanation I_T. This can be seen as an architectural change in the form of a branch. We take $F(x)$ to be a randomly initialized CNN (only the convolutional part). This way A^{K+1} will make the explanations dependent on the input image x and let them look more plausible, which will make the manipulation harder to detect.

To ensure large α_{K+1}^{c}, we add a branch from A^{K+1} to y. $\mathbb{1}$ is a vector of all ones. We set

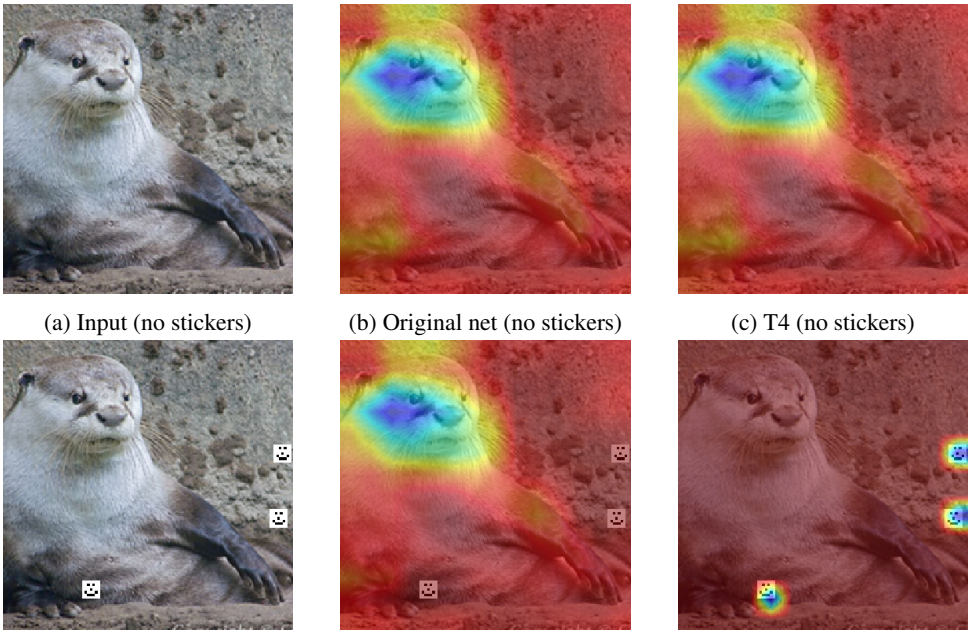
$$y_n = y_o + \mathbb{1} G(\text{flatten}(A_n^{K+1})).$$

$G(v)$ is a scalar valued function taking a vector of length N_A as input. We choose

$$G(v) = \epsilon \bmod (c_G \sum_i v_i, 1),$$

where $mod(a, b)$ is the modulus operator ensures that $G(v) \leq \epsilon$ for all v. By choosing ϵ to be small, the difference between the scores will be small: $|y_n - y_o| \leq \epsilon$. Furthermore, for all

inputs x we have $\frac{\partial G(x)}{\partial x} = \mathbb{1}c_G\epsilon$. By choosing $c_G \gg \epsilon$, we can make the gradient as large as desired, ensuring α_c^{K+1} will be large for all classes c.



(a) Input (no stickers) (b) Original net (no stickers) (c) T4 (no stickers)

(d) Input (stickers) (e) Original network (stickers) (f) T4 (stickers)

Figure 2.3.: Illustration of Technique 4. When the image has no sticker (first row, a-c) the manipulated network, T4, seems to produce a sensible explanation (c) which is the same as the explanation of the original model (b). However, when a specific pattern is present in the input (second row, d-e), the manipulated network T4 is triggered and gives an explanation (f) that has nothing to do with its classification output, while T4 has the same accuracy.

2.3.4. TECHNIQUE 4: MALICIOUS EXPLANATION TRIGGERED BY INPUT PATTERN

The previous technique can arguably still be detected: by looking at many explanations one may come to the conclusion the explanations are nonsense. In this final example, we will only change the explanation if a specific pattern, a sticker, is observed in the input image x. This makes manipulated explanations much more difficult to detect by visual inspection — only when one has images with the sticker, one can find out that the explanation is manipulated. See Figure 2.3.

We use exactly the same setup as in Technique 3, except that we change $F(x)$. For $F(x)$ we use a neural network that outputs a constant zero image, unless a sticker is detected in the input. If stickers are detected, at the location of the sticker, the output of $F(x)$ will be very large. Therefore, if no stickers are present, the explanation of the original network will

be returned, and if stickers are visible, the explanation will point at the stickers. Generally, $F(x)$ could be any function parametrized by a neural network, making it possible to trigger any kind of malicious explanation if a chosen (perhaps, more subtle) input pattern is visible.

2.4. Experimental Setup

For all experiments, we use the VGG-16 network [35]. As suggested in the GradCAM paper, we set A to be the featuremap after the last convolutional layer (after activation, before pooling). For VGG-16, $K = 512$ and the resolution of A^k is 14×14. We evaluate the original network and manipulated networks on the validation set of Imagenet of the ILSVRC2012 competition [36]. We generate the heatmap for the class with the highest posterior. The heatmap I^c always has positive values due to the ReLU operation. We normalize all heatmaps by the largest value in the heatmap to map it to $[0, 1]$: $\tilde{I}^c = \frac{I^c}{\max_{i,j} I^c_{ij}}$. We measure to what extent our manipulations are successful by measuring the distance between our target explanation $\tilde{I}_T$ and manipulated explanation $\tilde{I}_n$ in terms of the L_1 distance. For the experiments with the network T4, we evaluate on the original Imagenet validation set and the manipulated validation set. Manipulated images have 3 randomly placed smileys.

For T1, set $c_A = 100$, $c_W = 100$. For T2, set $c_W = 10$ and we set I_T to a 14×14 smiley image. For T3, choose $\epsilon = 0.01$, $c_G = 10000$ and $c_F = 1E7$. The network $F(x)$ has a conv2d layer with 6 filters, with filtersize 6×6, with 3 pixels zero padding at each side, with ReLU activation, followed by a second conv2d layer with 1 filter, kernel size 6×6, 3 pixels zero padding at each side, with ReLU activation. All weights are randomly initialized. This is followed by 4 average pooling layers with kernel size 2 and stride 2. Then the output of $F(x)$ is 14×14 and, thus, matches the size of A^{K+1} for VGG-16. For T4 we use a network $F(x)$ that has only one conv2d layer. The smiley pattern is binary: each pixel is white or black. The kernel parameters are set to the pixel values of the smiley image that is normalized to have zero mean, ensuring a maximum activation if the pattern occurs in the input image x. We set the bias of the convolutional layer to $b = -\sum_{ij} I^2_{ij}(1 - \frac{1}{N}\sum_{ij} I_{ij}) + 0.0001$ where I_{ij} are the pixel values of the non-normalized smiley image. If the pattern is detected the output is 0.0001, typically otherwise the output will be negative. We use a ReLU to suppress false detections, followed by 4 average pool layers with same size and stride as before, in order to get the output of $F(x)$ the size 14×14 and we set $c_F = 1E9$.

2.5. Results

The results for techniques T1-T3 are shown in Table 2.1, for qualitative results see Figure 2.1. A minimal change in accuracy and scores is observed. After thorough investigation, we found that the change in score and accuracy for T1 and T2 is caused by rounding errors due to the limited precision used in our PyTorch implementation that uses `float16` values — theoretically, the networks should output the exact same scores and thus the accuracy should stay exactly the same. The L_1 distance between our desired target explanation and our observed manipulated explanation is quite small, which matches with the qualitative observation in Figure 2.1. Note that the change in score for T3 is lower than ϵ, as guaranteed.

The results for technique T4 are shown in Table 2.2, for a qualitative example see Figure 2.3. We observe a small drop in accuracy when the data is manipulated by stickers, as expected, but the accuracy for T4 and the original network are exactly the same. The change in score is very small. If there are no stickers, the target explanation $\tilde{I}_T$ is equal to the explanation of the original network. If there are stickers, $\tilde{I}_T$ is equal to the heatmap that detects the stickers. The observed explanation when a sticker is present is almost equal to the target explanation. Just as desired, if no sticker is present, the explanation of T4 remains the same as the explanation of the original network.

	Accuracy	$\|\|y_o - y_n\|\|_\infty$	$\|\|\tilde{I}_T - \tilde{I}_n\|\|_1$
Original network	0.71592	-	-
T1: constant	0.71594	0.01713	0.00513
T2: smiley	0.71594	0.00454	0.01079
T3: random	0.71592	0.00000	0.05932

Table 2.1.: Evaluation of manipulated networks T1-T3 on the ILSVRC2012 validation set. Observe that the accuracy more or less stays the same. We measure the difference between the score y_o of the original network and new manipulated score y_n (the score is the output before softmax). The difference between the desired target explanation $\tilde{I}_T$ and the actual observed explanation $\tilde{I}_n$ is measured using the L_1 distance. The score changes very little while we can accurately manipulate the explanation as indicated by small L_1 distance.

Dataset	Network	Accuracy	$\|\|y_o - y_n\|\|_\infty$	$\|\|\tilde{I}_T - \tilde{I}_n\|\|_1$
Original	Original	0.71592	-	-
	T4: backdoor	0.71592	0.00000	0.00000
Manipulated (sticker)	Original	0.69048	-	-
	T4: backdoor	0.69048	0.00000	0.00006

Table 2.2.: Evaluation of Technique 4 on the ILSVRC2012 validation set. Observe that T4 has the same accuracy and scores as the original network for both kinds of data. When presented with input data without stickers, the manipulated network T4 produces the same explanation as the original network. When presented with manipulated data, the manipulated explanation, $\tilde{I}_n$, is almost equal to the desired explanation, $\tilde{I}_T$.

2.6. DISCUSSION

GradCAM is not 'broken' — for normally trained models, GradCAM has been proven to be useful. GradCAM does not work for adverserially manipulated models such as ours, since it was not designed for that task. However, our models are valid models, with (almost) equal performance. Hence, they should also admit a valid explanation. In fact, in [7] the axiom of Implementation Invariance is defined: two networks that produce the same output for all inputs should admit the same explanation. Clearly, GradCAM does not satisfy this axiom and thus there is room for improvement. One may wonder wether the axiom should be extended to models that return extremely similar predictions, such as T3 and T4.

Our work reveals that GradCAM relies on unknown assumptions on the network parameters, architecture, etc. It is difficult to rule out that, by accident, a model can be produced, using regular training, where GradCAM explanations may fail. We think it is important to determine what assumptions should be verified for GradCAM to produce accurate explanations, so we can always verify the correctness of GradCAM explanations.

Our techniques may be extended to fool other explanation methods. Several methods rely on the gradient $\frac{\partial y}{\partial x}$ [3, 7, 11, 13, 14]. T3 and T4 show that it is possible to manipulate the gradient, while affecting accuracy only little. So, these methods may also be vulnerable.

A weakness of our method is that architectural changes are necessary. If the practitioner visualizes the architecture (for example, using TensorBoard in TensorFlow [37]) or inspects the code, he may easily discover that the model has been tampered with. However, we believe similar attacks, where the original architecture is used, should be feasible, which would make the attack much harder to detect. We believe this is possible, since deep networks contain a lot of redundancy in the weights. Weights can be compressed or pruned, freeing up neurons, which then may be used to confuse the explanation. Recently, this area of research has been very active [38, 39]. For example, Srinivas and Babu [40] were able to prune 35% of the weights, while not significantly changing the test accuracy on MNIST. Another approach is Knowledge Distillation (KD), where a larger model (the teacher) can be compressed in a smaller model (the student) [41]. Such methods could be combined with our technique to keep the model accuracy more or less the same and to confuse the explanation method, without any architectural changes. We will explore this promising idea in future work.

2.7. CONCLUSION

We provided another sanity check in the same vein as Adebayo *et al.* [21] and we have shown that GradCAM does not satisfy said sanity check. We submit that, for any explanation method, one should consider whether it is possible to change the underlying model such that the predictions change minimally, while explanations change significantly. If this is the case, our work illustrates that the explanation method may be fooled by an attacker with access to the model and the explanations may not be as robust as desired.

2.8. BIBLIOGRAPHY

[1] T. Viering, Z. Wang, M. Loog, and E. Eisemann, *How to manipulate cnns to make them lie: the gradcam case,* arXiv preprint arXiv:1907.10901 (2019).

[2] A. Adadi and M. Berrada, *Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI),* IEEE Access **6**, 52138 (2018).

[3] K. Simonyan, A. Vedaldi, and A. Zisserman, *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps,* in *2nd International Conference on Learning Representations (ICLR), Banff, AB, Canada, Workshop Track Proceedings* (2013).

[4] S. Lundberg and S.-I. Lee, *A unified approach to interpreting model predictions,* in *Proceedings of Advances in Neural Information Processing Systems 30 (NIPS)* (2017) pp. 4768–4777.

[5] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, *Grad-cam: Visual explanations from deep networks via gradient-based localization,* in *Proceedings of the IEEE International Conference on Computer Vision* (2017) pp. 618–626.

[6] P.-J. Kindermans, K. T. Schütt, M. Alber, K.-R. Müller, D. Erhan, B. Kim, and S. Dähne, *Learning how to explain neural networks: PatternNet and PatternAttribution,* in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada* (2018).

[7] M. Sundararajan, A. Taly, and Q. Yan, *Axiomatic Attribution for Deep Networks,* in *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (2017) pp. 3319–3328.

[8] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, *Striving for Simplicity: The All Convolutional Net,* in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, Workshop Track Proceedings* (2014).

[9] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks,* in *European Conference on Computer Vision (ECCV)* (2014) pp. 818–833.

[10] P. W. Koh and P. Liang, *Understanding black-box predictions via influence functions,* in *Proceedings of the 34th International Conference on Machine Learning* (2017) pp. 1885–1894.

[11] A. Shrikumar, P. Greenside, and A. Kundaje, *Learning Important Features Through Propagating Activation Differences,* in *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (2017) pp. 3145–3153.

[12] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, *On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation,* PloS one **10**, e0130140 (2015).

[13] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, *SmoothGrad: removing noise by adding noise,* arXiv preprint arXiv:1706.03825 (2017).

[14] M. T. Ribeiro, S. Singh, and C. Guestrin, *"why should i trust you?": Explaining the predictions of any classifier,* in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2016) pp. 1135–1144.

[15] M. L. Amogh Gudi Nicolai van Rosmalen and J. van Gemert, *Object-Extent Pooling for Weakly Supervised Single-Shot Localization,* in *Proceedings of the British Machine Vision Conference (BMVC)* (2017).

[16] R. C. Fong and A. Vedaldi, *Interpretable Explanations of Black Boxes by Meaningful Perturbation,* Proceedings of the IEEE International Conference on Computer Vision **2017-Octob**, 3449 (2017).

[17] P. Dabkowski and Y. Gal, *Real time image saliency for black box classifiers,* in *Proceedings of Advances in Neural Information Processing Systems 30 (NIPS)* (2017) pp. 6967–6976.

[18] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, *Visualizing Deep Neural Network Decisions: Prediction Difference Analysis,* in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France* (2017).

[19] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, *Object Detectors Emerge in Deep Scene CNNs,* in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA* (2015).

[20] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, *A unified view of gradient-based attribution methods for Deep Neural Networks,* in *NIPS Workshop on Interpreting, Explaining and Visualizing Deep Learning* (2017).

[21] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, *Sanity checks for saliency maps,* in *Proceedings of Advances in Neural Information Processing Systems 31 (NIPS)* (2018) pp. 9505–9515.

[22] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim, *The (Un)reliability of saliency methods,* arXiv preprint arXiv:1711.00867 (2017).

[23] A. Ghorbani, A. Abid, and J. Zou, *Interpretation of Neural Networks is Fragile,* arXiv preprint arXiv:1710.10547 (2017).

[24] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, *Learning Deep Features for Discriminative Localization,* in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[25] G. Inc., *Google Cloud Machine Learning Engine,* (2019).

[26] A. Inc., *Amazon SageMaker,* (2019).

[27] M. Inc., *Azure Machine Learning Service,* (2019).

[28] Q. Yao, M. Wang, Y. Chen, W. Dai, H. Yi-Qi, L. Yu-Feng, T. Wei-Wei, Y. Qiang, and Y. Yang, *Taking Human out of Learning Applications: A Survey on Automated Machine Learning,* arXiv preprint arXiv:1810.13306 (2018).

[29] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, *CNN features off-the-shelf: An astounding baseline for recognition,* IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop, Columbus, OH, USA , 512 (2014).

[30] T. Gu, B. Dolan-Gavitt, and S. Garg, *BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain,* arXiv preprint arXiv:1708.06733 (2017).

[31] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, *Trojaning attack on neural networks,* in *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS)* (2018).

[32] B. Wang, Y. Yuanshun, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, *Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks,* in *2019 IEEE Symposium on Security and Privacy (SP)* (2019).

[33] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava, *Detecting backdoor attacks on deep neural networks by activation clustering,* Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19) (2019).

[34] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, *Towards the Science of Security and Privacy in Machine Learning,* arXiv preprint arXiv:1611.03814 (2016).

[35] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition,* in *International Conference on Learning Representations* (2015).

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and others, *Imagenet large scale visual recognition challenge,* International journal of computer vision **115**, 211 (2015).

[37] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, *Tensorflow: Large-scale machine learning on heterogeneous systems,* (2015).

[38] J. Cheng, P. Wang, G. Li, Q. Hu, and H. Lu, *Recent Advances in Efficient Computation of Deep Convolutional Neural Networks,* Frontiers of Information Technology & Electronic Engineering **19**, 64 (2018).

[39] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, *A Survey of Model Compression and Acceleration for Deep Neural Networks,* arXiv preprint arXiv:1710.09282 (2017).

[40] S. Srinivas and R. V. Babu, *Data-free Parameter Pruning for Deep Neural Networks,* in *Proceedings of the British Machine Vision Conference (BMVC)*, edited by Xianghua Xie Mark W. Jones and G. K. L. Tam (2015) pp. 1–31.

[41] J. Ba and R. Caruana, *Do deep nets really need to be deep?* in *Proceedings of Advances in Neural Information Processing Systems 27 (NIPS)* (2014) pp. 2654–2662.

3

NUCLEAR DISCREPANCY FOR SINGLE-SHOT BATCH ACTIVE LEARNING

Active learning algorithms propose what data should be labeled given a pool of unlabeled data. Instead of selecting randomly what data to annotate, active learning strategies aim to select data so as to get a good predictive model with as little labeled samples as possible. Single-shot batch active learners select all samples to be labeled in a single step, before any labels are observed. We study single-shot active learners that minimize generalization bounds to select a representative sample, such as the Maximum Mean Discrepancy (MMD) active learner. We prove that a related bound, the Discrepancy, provides a tighter worst-case bound. We study these bounds probabilistically, which inspires us to introduce a novel bound, the Nuclear Discrepancy (ND). The ND bound is tighter for the expected loss under optimistic probabilistic assumptions. Our experiments show that the MMD active learner performs better than the Discrepancy in terms of the mean squared error, indicating that tighter worst case bounds do not imply better active learning performance. The proposed active learner improves significantly upon the MMD and Discrepancy in the realizable setting and a similar trend is observed in the agnostic setting, showing the benefits of a probabilistic approach to active learning. Our study highlights that assumptions underlying generalization bounds can be equally important as bound-tightness, when it comes to active learning performance.

This work was presented at ECML PKDD 2019, Wurzburg, Germany and was published in the proceedings of Machine Learning [1]. Appendix C on page 163 gives background knowledge, proofs, additional remarks and experiments.

3.1. INTRODUCTION

Supervised machine learning models require enough labeled data to obtain good generalization performance. For many practical applications such as medical diagnosis or video topic prediction labeling data can be expensive or time consuming [2]. Often in these settings unlabeled data is abundant. In active learning an algorithm chooses unlabeled samples for labeling [3]. Models can perform better with less labeled data if the labeled data is chosen carefully instead of randomly. Active learning aims to makes the most of a labeling budget and can reduce labeling costs.

Several works use upperbounds on the expected loss to motivate particular active learning strategies [4–8]. We study pool-based active learners that choose queries that explicitly minimize generalization bounds and investigate the relation between bounds and active learning performance. We evaluate generalization with respect to the surrogate loss in the classification setting and use the kernel regularized least squares model [9], a popular model in active learning [8, 10]. Our focus is on active learners that select a batch of queries in a single shot [11]. This means that there is no label information available at the time the batch of queries is determined. Since the active learners have only have unlabeled data at their disposal they aim to select the most representative subset of the unlabeled pool. This is different from batch mode or sequential active learning, where after requesting labels from the oracle the algorithm has to determine new queries, creating a feedback loop. The advantage of zero-shot active learning is that all queries can be computed ahead of time, and collected labels do not have to be fed into the active learner.

For applications this can be very convenient: it simplifies the annotation setup. Furthermore, active learning algorithm may require substantial amounts of time to compute the next query. In situations where annotation have to be done by domain experts whose time is costly this can be impractical. For example, if we were to apply active learning to to the problem of Esteva *et al.* [12], who build a deep learning model to classify skin cancer, sequential or batch mode active learning strategies usually train a model as intermediate step before being able to determine the next query. For deep models this could take several hours. With zero-shot active learning the dermatologist can annotate all queries without waiting once.

Another example where requesting labels is costly is personalized machine learning models such as for movie recommendation. Here applications may ask feedback from end-users to improve their service. This problem can also be studied using the active learning framework [13]. Asking end-users for feedback usually interrupts their activity in the application. Therefore, we may only interrupt the user a limited amount of times. Using zero-shot active learning users only have to be interrupted once and can answer multiple queries without waiting for new queries.

The Maximum Mean Discrepancy (MMD) is used for batch-mode active learning by Chattopadhyay *et al.* [14] to match the marginal distribution of the selected samples to the marginal distribution of all unlabeled samples. This active learner has been shown to minimize a generalization bound [8]. The MMD is a divergence measure [15] which is closely related to the Discrepancy divergence measure of Mansour *et al.* [16], both have been used in domain adaptation [17, 18].

Using the Discrepancy, we show that we can get a tighter worst case generalization bound than the MMD in the realizable setting. Tighter bounds are generally considered better as they estimate the expected loss more accurately. One might therefore expect the Discrepancy to lead to better queries in active learning.

We show, however, that the Discrepancy and MMD generalization bounds can be derived, using a probabilistic analysis, from pessimistic assumptions. We subsequently apply the principle of maximum entropy to derive probabilistic assumptions that are more optimistic, inspiring us to introduce the Nuclear Discrepancy (ND) bound. Under these optimistic assumptions the ND provides a tighter bound on the expected loss than the MMD, while the Discrepancy bound is the loosest.

We compare the active learning performance of the proposed ND bound to the existing MMD and Discrepancy bounds. Our hypothesis is that we often find ourselves in a more optimistic average-case scenario than a worst-case scenarios. To this end we empirically study the behavior of the active learners on 13 datasets, and we investigate whether probabilistic assumptions or worst-case assumptions better model observed behavior in our experiments.

In the realizeable setting a model from the model class can perfectly predict the groundtruth labels, as in this setting there is no model mismatch or model misspecification. For this we show that the tightness relations between the generalization bounds is strict. As such, for the realizeable case, our theory gives the strongest predictions for the ranking of the active learners in terms of performance. In the agnostic case, where no such model may exist, the tightness relations can change, which renders our theory less applicable. We perform experiments in both settings to see the effect of the theoretical assumptions not being fulfilled.

We study the realizable setting since it is more amendable to theoretical analysis. This setting is often studied in active learning and is still a topic of active investigation [19]. The general case of the agnostic case is much harder to analyze. For example, it has been observed that if a model class is sufficiently wrongly chosen, active learning can even decrease model performance [20–23].

These counter-intuitive behaviors further underline the need for further theoretical studies. We believe that by improving our understanding of simpler active learning settings (realizeable case) will contribute to improved understanding of more difficult active learning settings (agnostic case).

To this end, our study provides new quantitative tightness relations between the MMD, Discrepancy and ND bound under different probabilistic assumptions. We investigates the connection between bound tightness and active learning performance. Our most important conclusion is that not only bound tightness is important for performance, but that appropriate assumptions are equally important.

3.1.1. OVERVIEW AND CONTRIBUTIONS

First we discuss related work in Section 3.2. In Section 3.3 we describe the considered active learning setting and notation. We present our theoretical results regarding the MMD and Discrepancy in Section 3.4. In Section 3.5 we motivate our novel Nuclear Discrepancy bound. We evaluate the proposed active learners experimentally in Section 3.6. In Section 3.7 we give a discussion and in Section 3.8 we give the conclusions of this work. All proofs, additional background theory and experimental results are given in Appendix C (page 163). The main contributions are:

1. An improved MMD bound for active learning and a more informed way to choose the kernel of the MMD in the context of learning.
2. A proof that the Discrepancy bound on the worst case loss is tighter than the MMD bound.
3. A probabilistic interpretation of the MMD bound.
4. The Nuclear Discrepancy (ND) bound that provides the tightest bound on the expected loss under probabilistic assumptions that follow from the principle of maximum entropy.
5. A probabilistic analysis that explains the differences in empirical performance (in terms of the mean squared error) achieved by the active learners.

In Table 3.1 we give a visual summary of our work. It shows all formal results and shows in which sections to find them. It also shows the relation between the theory and experiments, and the main findings of the experiments.

3.2. RELATED WORK

Many active learning methods have been proposed, Settles [2] provides an excellent introduction and overview. Our work is related to active learning methods that select representative samples [24]. Most active learning strategies of this kind are combined with an uncertainty criteria [8, 10, 14, 24], and often the representative component is used to diversify queries when chosen in batches in order to avoid redundancy [8, 24]. This is different from our considered setting: since there is no labeled data and we have to choose all queries in one shot, our only option is to select representative samples, since uncertainty criteria can only be computed if some labels are known.

A closely related well-known concept to our work is that of (Transductive or) Optimal Experimental Design [25]. Here also no labeled data is required to select queries for the case of the linear regression model. These methods aim to minimize some form of posterior variance of the model. A closely related statistical approach relies on maximization of the Fisher Information to reduce model uncertainty [26]. However, for these approaches it is often required to explicitly specify a noise model (such as Gaussian i.i.d. noise), while in this work we consider deterministic labels.

Our work is motivated by several active learners that minimize generalization bounds. Gu and Han [4] uses the Transductive Rademacher Complexity generalization bound to

Table 3.1.: Visual summary of our work. This table gives an overview of the newly proven tightness relations between the generalization bounds and the experimental results. Observe that the tightness relations under the 'Average-Case' correlate well with the experimental performance of the active learners. Therefore, we stipulate that the 'Average-Case' is the most accurate assumption for our considered active learning setting. Note that the tightness relations only hold under the conditions of Theorem 2, and that the experimental performance shown here best reflect the performance in the realizable setting. In the agnostic setting the ranking of the active learning methods is less clear, but the same trend is observed.

	Probabilistic Assumption			Experiments
	Worst-Case	Pessimistic-Case	Average-Case	Performance
Bound	Section 3.4.3	Section 3.4.4	Section 3.5	Section 3.6
Discrepancy	Tightest	Loosest	**Loosest**	**Worst**
MMD	Intermediate	Tightest	**Intermediate**	**Intermediate**
Nuclear Discrepancy (proposed)	Loosest	Intermediate	Tightest	Best

perform active learning on graphs. Gu *et al.* [5] show that the strategy of Yu *et al.* [25] also minimizes a generalization bound, and extend the method to work with a semi-supervised model. Ganti and Gray [6] introduce an active learning strategy that uses importance weighting to ensure asymptotic consistency of the actively learned model. Their strategy minimizes a generalization bound for the squared loss under some conditions on the data distribution. Gu *et al.* [7] introduce an strategy that minimizes a generalization bound on the risk for logistic regression. Wang and Ye [8] also use a generalization bound based on the MMD to perform active learning, but we will describe this work later in more detail when discussing all methods that use the MMD.

Many theoretical active learning works motivate algorithms by generalization bounds, for example one of the first active learning algorithms 'CAL' [3] and its agnostic generalization A^2 [27] have been thoroughly analyzed using generalization bounds by making use of the Disagreement Coefficient [28]. Most of these theoretical works consider worst-case performance guarantees, where the distribution is chosen by an adversary subject to constraints. Balcan and Urner [29] provides a short and concise overview of these and other recent theoretical active learning works. In contrast with our work, these algorithms consider generalization in terms of zero-one loss instead of squared loss and do not apply to one shot active learning.

A straightforward approach to one shot active learning is through clustering: cluster the data and request the labels of the cluster centers [30–33]. However, unlike our work,

these methods are not motivated by generalization bounds. Obtaining bounds for such approaches may be difficult because the clustering algorithm and machine learning model may rely on different assumptions. To still get bounds one can use the clustering algorithm instead to also provide predictions for new samples [34]. Instead, we stick to the regularized least squares model and use the MMD and Discrepancy to get bounds for this model. Our approach can be used to derive bounds and corresponding active learning strategies for any kernelized L_2 regularized model, however, in this work we only focus on the squared loss.

Our work is closely related to that of Chattopadhyay *et al.* [14]: we use a greedy version of their proposed active learning algorithm. Chattopadhyay *et al.* [14] are the first to use the MMD for active learning in a batch-mode setting. An in-depth empirical analysis shows that the MMD outperforms other active learning criteria as judged by the zero-one error when used with kernelized SVMs. They show that the MMD easily can be combined with uncertainty-based active learning approaches and transfer learning. Since we consider one-shot active learning we do not consider the uncertainty-based component of their algorithm. In follow up work active learning and transfer learning is solved jointly using the MMD [35].

Our theoretical analysis of the MMD bound extends the analysis of Wang and Ye [8]. Wang and Ye [8] show that active learning by minimization of the MMD and the empirical risk can be seen as minimizing a generalization bound on the true risk. They introduce an active learner that balances exploration (distribution matching using MMD) with exploitation (a form of uncertainty sampling). They show empirically that their proposed algorithm is competitive with several other active learning strategies as evaluated by the zero-one error using kernelized SVMs.

We build upon the generalization bound of Wang and Ye [8] and improve it. Their bound considers the underlying distribution of the unlabeled pool and labeled (queried) sample, however, this is problematic because the labeled sample is non-i.i.d. due to dependence of the queries of the active learner. We resolve this issue and introduce an additional term η that measures the error of approximating the worst-case loss function.

Mansour *et al.* [16] introduce the Discrepancy generalization bound for domain adaptation with general loss functions. In a follow up work, Cortes and Mohri [18] contrast the Discrepancy with the MMD generalization bound: they argue that the Discrepancy is favorable from a theoretical point of view because it takes the loss function and hypothesis set of the model into account, while the MMD does not. This means that the MMD bound for an SVM and regularized least squares model would be exactly the same, while the Discrepancy bound specializes to the chosen model and surrogate loss. They derive an efficient domain adaptation algorithm and empirically show that the Discrepancy improves upon the MMD in several regression adaptation tasks.

Prior to our work, the Discrepancy measure [18] has not yet been used to perform active learning. We show that by choosing the kernel for the MMD carefully, we can adapt the MMD to take the hypothesis set and loss into account, addressing one of the theoretical limitations of the MMD identified by Cortes and Mohri [18]. Under these conditions we find that we can compare the MMD and Discrepancy bounds in terms of tightness. This quantitative comparison of these bounds is novel and was not considered before.

Germain *et al.* [36] adapt the Discrepancy for the zero-one loss to a PAC-Bayes setting in order to do domain adaptation. Their analysis is specifically for the zero-one loss, while we consider the squared loss. Their PAC-Bayes framework is significantly different from our analysis: instead of minimizing a surrogate loss, they use a Gibbs classifier, and they minimize bounds on the expected risk directly. This involves a non-convex optimization problem. Instead, we simply minimize the empirical risk and consider deterministic models, similar to most PAC style analysis. This makes our analysis is simpler. Furthermore, they propose a framework to jointly minimize the empirical risk and domain divergence. To this end, their algorithm requires labeled data which is unavailable in zero-shot active learning, making it unsuitable for our zero-shot setting.

In Cortes *et al.* [37] a new domain adaptation algorithm based on a new divergence measure, the Generalized Discrepancy, is introduced. The algorithm consists of two stages: first it minimizes the Discrepancy, afterward it minimizes the empirical risk and the Generalized Discrepancy jointly. The strategy of Cortes *et al.* [37] is difficult to apply to active learning for two reasons. First of all, their algorithm requires labeled data to minimize the empirical risk and the General Discrepancy jointly, which is impossible in our zero-shot active learning setting. Second, their algorithm requires i.i.d. samples from the unlabeled pool to estimate the hyperparameter r. This would require costly random queries in the active learning setting. Because of these reasons, we believe their algorithm is more suitable to a joint active and domain adaptation setting (such as considered by Chattopadhyay *et al.* [35]) where more labeled data is available.

Our theoretical analysis is substantially different from the analysis of Cortes *et al.* [37]. Because Cortes *et al.* [37] use labeled data, they can make a more accurate characterization of possible worst case scenario's, refining the worst-case scenario of the Discrepancy to obtain tighter bounds. We take an orthogonal approach: we consider probabilistic generalization bounds that hold in expectation. Instead of considering a worst-case, we make probabilistic assumptions to get to a plausible average-case. Cortes *et al.* [37] compare the Generalized Discrepancy and Discrepancy bounds in terms of tightness. We compare the tightness of the bounds of the MMD, Discrepancy and Nuclear Discrepancy. We show several orderings of the tightness of the bounds under different probabilistic assumptions, while Cortes *et al.* [37] *only* takes a worst-case approach.

In summary, our work differs from previous works by considering instead of worst-case analysis [18, 37], a probabilistic analysis of generalization bounds. Unlike most other works that use generalization bounds for domain adaptation [18, 36, 37], we use bounds to perform active learning. For the MMD active learner, studied by Wang and Ye [8], Chattopadhyay *et al.* [14], we give new theoretical results: an improved bound for active learning and we provide a principled way to choose the kernel for the MMD. We give new quantitative comparisons of bound tightness for the MMD and Discrepancy in multiple settings, while before these bounds were compared only qualitatively [18]. Furthermore, we study the novel question: how does bound tightness relate to active learning performance?

3.3. SETTING AND NOTATION

Let $\mathcal{X} = \mathbb{R}^d$ denote the input space and $\mathcal{Y}$ the output space. Like Cortes and Mohri [18] we assume there is a function $f : \mathcal{X} \to \mathcal{Y}$ that determines the outputs and there is an unknown

distribution with density P over $\mathscr{X}$ from which we get an independent and identically distributed (i.i.d.) unlabeled sample $\hat{P} = (x'_1, \ldots, x'_{n_{\hat{P}}}) \in \mathscr{X}^{n_{\hat{P}}}$. We study single-shot batch active learners that given the unlabeled pool $\hat{P}$ selects a batch $\hat{Q}_n \subset \hat{P}$ of n samples before observing any labels. The active learner submits the batch to the labeling oracle that provides the labels of the batch. A kernel regularized least squares (KRLS) model is trained on $\hat{Q}_n^{\text{lab}}$, where lab indicates a labeled dataset.

We take the kernel of the model K to be positive semi-definite (PSD), and denote the reproducing kernel Hilbert space (RKHS) as $\mathscr{H}$ where $||h||_K$ denotes the norm in $\mathscr{H}$. A model corresponds to $h \in \mathscr{H}$ and is obtained by minimizing

$$L_{\hat{Q}}(h, f) + \mu ||h||_K^2$$

for $h \in \mathscr{H}$ when trained on $\hat{Q}^{\text{lab}}$, where we follow the convention of Cortes and Mohri [18]. $L_{\hat{Q}}(h, f)$ is the average empirical loss of h on $\hat{Q}$ with outputs given by f:

$$L_{\hat{Q}}(h, f) = \frac{1}{n_{\hat{Q}}} \sum_{x \in \hat{Q}} l(h(x), f(x)),$$

where $l : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a loss function. For KRLS l is the squared loss: $l(h(x), f(x)) = (h(x) - f(x))^2$, then $L_{\hat{Q}}(h, f)$ is the mean squared error (MSE) on $\hat{Q}$. Model complexity is controlled by the regularization parameter $\mu > 0$. We choose

$$H = \{h \in \mathscr{H} : ||h||_K \leq \Lambda = \frac{f_{\max}}{\sqrt{\mu}}\}$$

as our hypothesis set where $f_{\max} = \sup_{x \in \mathscr{X}} |f(x)|$. Training KRLS always leads to a solution $h \in H$ [38, Lemma 11.1].

In classification typically we are interested in the zero-one error (accuracy), however, our study focuses on the squared loss (the surrogate loss). We use the squared loss because we can relate the bounds of the MMD, Nuclear Discrepancy and Discrepancy in closed form and compare them quantitatively. Since our goal is to investigate the correlation between bound tightness and performance, this is essential to our study.

We have made the standard assumption that the data comes from an unknown distribution P. The goal of the active learner is to choose a batch of queries in such a way as to minimize the expected loss of the model under this distribution P:

$$L_P(h, f) = \int_{\mathscr{X}} (h(x) - f(x))^2 P(x) dx. \tag{3.1}$$

Ideally we would want to train our model on $\hat{P}^{\text{lab}}$, since small $L_{\hat{P}}(h, f)$ will lead to small $L_P(h, f)$ if the model complexity is appropriate, as illustrated by the following theorem [38, p. 240].

Theorem 1 (Generalization bound Squared Loss[38])**.** *Let l be the squared loss. For any $\delta > 0$, with probability at least $1 - \delta$ over an i.i.d. sample $\hat{P}$ of size $n_{\hat{P}}$ from P, the following inequality holds for all $h \in H$:*

$$L_P(h, f) \leq L_{\hat{P}}(h, f) + 4MR_m(H) + M^2 \frac{\log(\frac{1}{\delta})}{2n_{\hat{P}}} \tag{3.2}$$

Here $R_m(H)$ is the Rademacher complexity of the hypothesis set H, and M is a constant such that $|h(x) - f(x)| \leq M$ for all $x \in \mathcal{X}$ and all $h \in H$.

If the model complexity is appropriate $R_m(H)$ will be small. The third term is small when the pool $\hat{P}$ is large. If both of these criteria are met, it is unlikely that we overfit as reflected by a tight bound. Then training on $\hat{P}^{\text{lab}}$ will likely minimize $L_P(h, f)$.

Ideally we would train on $\hat{P}^{\text{lab}}$, however, since we only have access to the unlabeled sample $\hat{P}$ this is impossible. Therefore we upperbound $L_{\hat{P}}(h, f)$ instead. This upperbound is minimized by the active learners. The studied bounds are of the form

$$L_{\hat{P}}(h, f) \leq L_{\hat{Q}}(h, f) + \text{obj}(\hat{P}, \hat{Q}) + \eta.$$

Due to training $L_{\hat{Q}}(h, f)$ will be relatively small. The term η is a constant that cannot be minimized during active learning since it depends on $\hat{P}^{\text{lab}}$. However, if the model misspecification is small, η will be small. Therefore we ignore this term during active learning, this is also (sometimes implicitly) done in other works [14, 17, 18]. Thus the active learners choose the batch $\hat{Q}$ to minimize $\text{obj}(\hat{P}, \hat{Q})$. This objective can be the MMD, disc or disc_N which will be introduced in the next sections. This term measures the similarity between the unlabeled pool $\hat{P}$ and the batch $\hat{Q}$. Minimizing it leads to selecting a representative sample.

We consider two settings. In the agnostic setting binary labels are used, i.e., $\mathcal{Y} = \{-1, +1\}$, and generally we have $f \notin H$. In the realizable setting $f \in H$, so a model of our hypothesis set can perfectly reproduce the labels as there is no model misspecification. In this case $\mathcal{Y}$ is a subset of $\mathbb{R}$. In the realizeable setting η can become zero under some conditions, which allows us to compare the tightness of the bounds and enables our probabilistic analysis.

$K(x, x')$ indicates the kernel function between x and x'. We mainly use the Gaussian kernel $K(x, x') = \exp(-||x - x'||_2^2/(2\sigma^2))$ where σ, the bandwidth, is a hyperparameter of the kernel. For the MMD we require a second PSD kernel, $K_{\mathcal{L}}$. We indicate its RKHS and bandwidth (for a Gaussian kernel) by $\mathcal{H}_{\mathcal{L}}$ and $\sigma_{\mathcal{L}}$, respectively. All vectors are column vectors. $X_{\hat{P}}$ and $X_{\hat{Q}}$ are the $n_{\hat{P}} \times d$ and $n_{\hat{Q}} \times d$ matrices of the sets $\hat{P}$ and $\hat{Q}$.

3.4. ANALYSIS OF EXISTING BOUNDS

First we provide an improved MMD generalization bound for active learning which is inspired by Cortes *et al.* [37]. Then we review a bound in terms of the Discrepancy of Cortes *et al.* [37] and we review how to compute the Discrepancy quantity [16]. We show that the MMD can be computed using a novel eigenvalue analysis, and thereby making the MMD and Discrepancy bounds comparable. We wrap up the section with a probabilistic interpretation of both bounds. As a roadmap for the reader we give an overview of the tightness relations in Table 3.1 which will be proven in this section and the next section.

3.4.1. IMPROVED MMD BOUND FOR ACTIVE LEARNING

The MMD measures the similarity between the two unlabeled samples $\hat{Q}$ and $\hat{P}$. Using this criterion we give a generalization bound similar to the one given by Wang and Ye [8] suitable for active learning. The empirical MMD quantity is given by

$$\mathrm{MMD}(\hat{P},\hat{Q}) = \max_{\tilde{l}\in H_{\mathscr{L}}}\left(\frac{1}{n_{\hat{P}}}\sum_{x\in\hat{P}}\tilde{l}(x) - \frac{1}{n_{\hat{Q}}}\sum_{x\in\hat{Q}}\tilde{l}(x)\right).$$

Here $\tilde{l}$ is the worst-case function from a set of functions $H_{\mathscr{L}}$. We take the standard choice $H_{\mathscr{L}} = \{h \in \mathscr{H}_{\mathscr{L}} : ||h||_{K_{\mathscr{L}}} \leq \Lambda_{\mathscr{L}}\}$. In Appendix C.1.1 we revisit how to compute the MMD quantity. We extend the technique of Cortes *et al.* [37] to give a generalization bound in terms of the MMD. To get a bound for the MMD we approximate the loss function $g(h,f)(x) = l(h(x),f(x))$ using $H_{\mathscr{L}}$.

Proposition 1 (Agnostic MMD worst case bound)**.** *Let l be any loss function $l : \mathbb{R}\times\mathbb{R}\to\mathbb{R}$. Then for all $h \in H$ and any labeling function $f : \mathscr{X}\to\mathscr{Y}$ we have*

$$L_{\hat{P}}(h,f) \leq L_{\hat{Q}}(h,f) + MMD(\hat{P},\hat{Q}) + \eta_{MMD}, \tag{3.3}$$

where $\eta_{MMD} = 2\min_{\tilde{l}\in H_{\mathscr{L}}}\max_{h\in H, x\in\hat{P}}|g(h,f)(x) - \tilde{l}(x)|$.

Here η_{MMD} measures the approximation error since we may have that $g(h,f) \notin H_{\mathscr{L}}$. Our MMD bound above differs in two aspects from the bound of Wang and Ye [8]. Wang and Ye [8] estimate the MMD between the distributions P and Q. However, to estimate the MMD between distributions i.i.d. samples are required [15, Appendix A.2]. The sample $\hat{Q}$ is not i.i.d. since it is chosen by an active learner.

Our bound allows for non-i.i.d. samples since it estimates the MMD between empirical samples and is therefore better suited for active learning. The second novelty is that we measure the error of approximating the loss function $g(h,f)$ using the term η_{MMD}. This allows us to adjust the MMD to the hypothesis set H and loss l similar to the Discrepancy measure of Cortes and Mohri [18]. We give the theorem below with a small proof sketch for the simplified case of the linear kernel. See the Appendix for the full proof.

Theorem 2 (Adjusted MMD)**.** *Let l be the squared loss and assume $f \in H$ (realizable setting). If $K_{\mathscr{L}}(x_i,x_j) = K(x_i,x_j)^2$ and $\Lambda_{\mathscr{L}} = 4\Lambda^2$, then $g(h,f) \in H_{\mathscr{L}}$ and thus $\eta_{MMD} = 0$.*

Proof sketch. Here we give a proof sketch for the case where K is the linear kernel: $K(x_i,x_j) = x_i^T x_j$. Then $h(x) = w_h^T x$ and $f(x) = w_f^T x$, and $g(h,f) = ((w_f - w_h)^T x)^2$ is a quadratic function of x. The featuremap of the kernel $K_{\mathscr{L}}(x_i,x_j) = K(x_i,x_j)^2$ are all monomials of degree 2 [39, chap. 9.1]. Therefore $H_{\mathscr{L}}$ can be used to model any quadratic function such as $g(h,f)$. Therefore if $\Lambda_{\mathscr{L}}$ is chosen appropriately we have $g(h,f) \in H_{\mathscr{L}}$.

Corollary 1. *Let l be the squared loss and $f \in H$ and let K be a Gaussian kernel with bandwidth σ. If $K_{\mathscr{L}}$ is a Gaussian kernel with bandwidth $\sigma_{\mathscr{L}} = \frac{\sigma}{\sqrt{2}}$ and $\Lambda_{\mathscr{L}} = 4\Lambda^2$ then $\eta_{MMD} = 0$.*

Compared to other works Theorem 2 gives a more informed way to choose the MMD kernel in the context of learning[1]. Typically, a Gaussian kernel is used for the MMD with $\sigma_{\mathscr{L}} = \sigma$. However, Corollary 1 shows that if $\sigma_{\mathscr{L}} = \sigma$, we may have that $\eta_{\text{MMD}} \neq 0$ even in the realizable setting, since $\sigma_{\mathscr{L}}$ is too large — the true loss function $g(h, f)$ is less smooth than the functions in $H_{\mathscr{L}}$. This is undesirable since η_{MMD} cannot be minimized during active learning. Our choice for $\sigma_{\mathscr{L}}$ is preferable, as it ensures $\eta_{\text{MMD}} = 0$ in the realizable setting.

3.4.2. DISCREPANCY BOUND

The Discrepancy is defined as

$$\text{disc}(\hat{P}, \hat{Q}) = \max_{h,h' \in H} |L_{\hat{P}}(h', h) - L_{\hat{Q}}(h', h)|. \tag{3.4}$$

Observe it depends on H and l and therefore automatically adjusts to the loss and hypothesis set. We give a bound of Cortes *et al.* [37] in terms of the Discrepancy.

Theorem 3 (Agnostic Discrepancy worst case bound[37])**.** *Assume that for all $x \in \mathscr{X}$ and for all $h \in H$ that $l(h(x), f(x)) \leq C$ and let l be the squared loss. Then for all $h \in H$ and any labeling function $f : \mathscr{X} \to \mathscr{Y}$ we have*

$$L_{\hat{P}}(h, f) \leq L_{\hat{Q}}(h, f) + disc(\hat{P}, \hat{Q}) + \eta_{disc},$$

where $\eta_{disc} = 4C \min_{\tilde{f} \in H} \max_{x \in \hat{P}} |\tilde{f}(x) - f(x)|$.

η_{disc} measures the model misspecification. In the realizable setting, $f \in H$, and $\eta_{\text{disc}} = 0$.

3.4.3. EIGENVALUE ANALYSIS

We show the relation between the Discrepancy and MMD using a novel eigenvalue analysis. To this end we introduce the matrix $M_{\hat{P},\hat{Q}}$ to compute the Discrepancy.

$$M_{\hat{P},\hat{Q}} = \frac{1}{n_{\hat{P}}} X_{\hat{P}}^T X_{\hat{P}} - \frac{1}{n_{\hat{Q}}} X_{\hat{Q}}^T X_{\hat{Q}},$$

For notational convenience we will often write M instead of $M_{\hat{P},\hat{Q}}$. The matrix M measures the difference between two sets of samples using their second-order moment. Considering its kernelized version such comparison can implicitly take higher-order moments into account as well. In particular, for a Gaussian kernel all moments of the samples are compared and we have that $M = 0$ only if $\hat{P} = \hat{Q}$.

In the following we will look at the eigendecomposition of M. Since M is the difference between two covariance matrices, it can have positive and negative eigenvalues. A positive (negative) eigenvalue means that in the direction of the corresponding eigenvector $\hat{P}$ has more (less) variance than $\hat{Q}$. Recall that in active learning, our aim is to approximate $\hat{P}$ using representative samples $\hat{Q}$, and thus small absolute eigenvalues are desirable, because this would indicate that in the direction of the corresponding eigenvector $\hat{P}$ is well approximated by $\hat{Q}$.

[1] The MMD is also used in other contexts, for example, the MMD can be used to determine if two sets of samples originate from the same distribution [15].

Theorem 4 (Discrepancy computation [16]). *Assume K is the linear kernel, $K(x_i, x_j) = x_i^T x_j$, and l is the squared loss, then*

$$disc(\hat{P}, \hat{Q}) = 4\Lambda^2 \max_i |\lambda_i| = 4\Lambda^2 ||\lambda||_\infty. \tag{3.5}$$

where λ_i are the eigenvalues of M, and λ is the vector of eigenvalues of M.

Note that h' will later play the role of f, the true labeling function. The theorem shows that in the worst case, the h and h' that maximize the Discrepancy in Equation 3.4 are chosen exactly in the direction where $\hat{Q}$ and $\hat{P}$ differ most, i.e., the direction of the largest absolute eigenvalue. Cortes and Mohri [18] show that we can replace M by M_K to compute the Discrepancy for any PSD kernel[2].

Before we can give our main result we require some additional notation. Assume that the eigenvalues λ_i of M are ordered by absolute value where $|\lambda_1|$ is the largest absolute eigenvalue. λ indicates the vector of eigenvalues, with $r = \text{rank}(M)$ non-zero eigenvalues. e_i is the normalized (unit-length) eigenvector corresponding to λ_i. By careful analysis we can realize the relationship between M and the featuremap of the squared kernel to show that the MMD can be computed as follows.

Theorem 5 (MMD Computation). *Let $K_{\mathscr{L}}(x_i, x_j) = K(x_i, x_j)^2$ and $\Lambda_{\mathscr{L}} = 4\Lambda^2$, then*

$$MMD(\hat{P}, \hat{Q}) = 4\Lambda^2 ||\lambda||_2. \tag{3.6}$$

This theorem shows that the MMD measures differences between the samples $\hat{Q}$ and $\hat{P}$ differently. The Discrepancy only measures similarity along one dimension, namely the direction where the samples differ the most. The MMD considers all dimensions to compare the samples $\hat{Q}$ and $\hat{P}$. Due to the square in the Euclidean norm, the MMD gives directions that differ more more weight in the comparison.

Corollary 2. *Under the conditions of Theorem 2, $disc(\hat{P}, \hat{Q}) \leq MMD(\hat{P}, \hat{Q})$.*

Under these conditions the Discrepancy bound (Theorem 3) is tighter than the MMD bound (Proposition 1), since $\eta_{\text{MMD}} = \eta_{\text{disc}} = 0$. Since the Discrepancy bound is tighter, one may expect that active learning by minimization of the Discrepancy may result in better active learning queries than minimization of the MMD, in particular if η_{MMD} and η_{disc} are small or zero.

3.4.4. PROBABILISTIC ANALYSIS

We show the MMD can provide a tighter bound on the expected loss under certain probabilistic assumptions. From this point on we assume the conditions of Theorem 2 and take h to be the model trained on the set $\hat{Q}$, and f to be the true labeling function. In addition, define $u = h - f$ and $U = \{u \in \mathscr{H} : ||u||_K \leq 2\Lambda\}$ and let $\bar{u}_i = u^T e_i$, where e_i is the eigenvector of M. Then $||u||_K = ||\bar{u}||_K \leq 2\Lambda$, since $\bar{u}$ is a rotated version of u. It is more convenient to work with $\bar{u}$, since then the matrix M diagonalizes: $u^T M u = \sum_i \bar{u}_i \lambda_i$.

[2] See the Appendix (Equation C.8) for the definition of M_K, additional details and the proof of this theorem. All our theoretical results that follow hold for both M and M_K. For simplicity we use M in the main text.

The difference u is the unknown error our trained model h makes compared with the true model f. By making different probabilistic assumptions about the distribution of u we can arrive at different bounds. We now provide the building block for our probabilistic bounds. By noting that $L_{\hat{P}}(h,f) - L_{\hat{Q}}(h,f) = u^T M u$ and by making use of the triangle inequality, we find the following.

Lemma 1 (Probabilistic bound)**.** *Assume*[3] *u is distributed according to a pdf $p(u)$ over U. Then*

$$\mathop{\mathbb{E}}_{u} L_{\hat{P}}(h,f) \leq \mathop{\mathbb{E}}_{u} L_{\hat{Q}}(h,f) + \mathop{\mathbb{E}}_{u} G(u,M), \tag{3.7}$$

where we defined $G(u,M) = \sum_i \bar{u}_i^2 |\lambda_i|$.

Observe that $G(u,M)$ is a weighted sum, where each $|\lambda_i|$ is weighted by $\bar{u}_i^2$. Recall that $L_{\hat{Q}}(h,f)$ is generally small due to the training procedure of the model, thus generally $\mathbb{E}_u L_{\hat{Q}}(h,f)$ will be small as well. Therefore we focus our probabilistic analysis on the term $\mathbb{E}_u G(u,M)$. By giving bounds on this quantity, we derive several probabilistic bounds that hold in expectation w.r.t. u.

The Discrepancy can be interpreted to put all probability mass on $u = 2\Lambda e_1$.

Proposition 2 (Worst case: Probabilistic Discrepancy)**.** *Given the pdf $p(u) = \delta(u - 2\Lambda e_1)$ where $\delta(x)$ is the Dirac delta distribution. Then*

$$\mathop{\mathbb{E}}_{u} L_{\hat{P}}(h,f) \leq \mathop{\mathbb{E}}_{u} L_{\hat{Q}}(h,f) + disc(\hat{P},\hat{Q}) \tag{3.8}$$

Only one $u \in U$ can be observed under this pdf. This is a worst case distribution because this $p(u)$ maximizes $\mathbb{E}_u G(u,M)$. The Discrepancy assumes that the model error u points exactly in the direction that causes us to make the biggest error on $\hat{P}$. Under this distribution the Discrepancy gives a tighter bound on the expected loss than the MMD because of Corollary 2. Under a different $p(u)$ the MMD bound is tighter.

Theorem 6 (Pessimistic case: Probabilistic MMD)**.** *Let $p(u)$ be a pdf on U_s such that* [4]

$$\mathop{\mathbb{E}}_{u} \bar{u}_i^2 = 4\Lambda^2 |\lambda_i| \left(\sqrt{r}||\lambda||_2\right)^{-1}, \tag{3.9}$$

then

$$\mathop{\mathbb{E}}_{u} L_{\hat{P}}(h,f) \leq \mathop{\mathbb{E}}_{u} L_{\hat{Q}}(h,f) + \frac{1}{\sqrt{r}} MMD(\hat{P},\hat{Q}) \leq \mathop{\mathbb{E}}_{u} L_{\hat{Q}}(h,f) + disc(\hat{P},\hat{Q}).$$

Unlike for the distribution of the Discrepancy, for the above $p(u)$ it is possible to observe different model errors u. However, the model error u in this case is biased: Equation 3.9 suggests that u is more likely to point in the direction of eigenvectors with large absolute eigenvalues. This assumption is pessimistic since large absolute eigenvalues can contribute

[3] This could be motivated for example, by placing a prior on f, then u would be a random variable. Another motivation is that we do not know u, and need to model it somehow to come to applicable generalization bounds. The Discrepancy assumes a worst-case scenario (it maximizes with respect to u), while we now consider assuming a distribution on u.

[4] To deal with infinite-dimensional RKHS we choose $p(u)$ on U_s instead of U, where U_s is the part of U restricted to the span of $X_{\hat{P}}$. Here r is the effective dimension: $r = \dim(U_s)$. This is necessary, otherwise sampling uniformly from an infinite-dimensional sphere can lead to problems. See Appendix C.3 for more details.

more to $\mathbb{E}_u G(u, M)$. Another way to interpret this is that model errors are more likely to occur in directions where $\hat{Q}$ and $\hat{P}$ differ more. Because $\hat{Q}$ and $\hat{P}$ differ more in those directions, these model errors can count more towards the MSE on $\hat{P}$.

For this $p(u)$ the MMD bound is tighter. If the probabilistic assumption of the MMD is more accurate, we can expect that the MMD active learner will yield better active learning queries than the Discrepancy.

3.5. NUCLEAR DISCREPANCY

In this section we motivate the optimistic probabilistic assumption that leads to the Nuclear Discrepancy (ND) bound. First, let us introduce the Nuclear Discrepancy quantity

$$\text{disc}_N(\hat{P}, \hat{Q}) = 4\Lambda^2 ||\lambda||_1.$$

In the absence of any prior knowledge, we choose the pdf $p(u)$ according to the well established principle of maximum entropy. This principle dictates that in case nothing is known about a distribution, the distribution with the largest entropy should be chosen [40]. Accordingly, we choose $p(u)$ uniform over U, which leads to the following.

Theorem 7 (Optimistic case: Probabilistic ND)**.** *Let $p(u)$ be uniform [4] over all $u \in U_s$, then*

$$\mathop{\mathbb{E}}_{u} L_{\hat{P}}(h, f) \leq \mathop{\mathbb{E}}_{u} L_{\hat{Q}}(h, f) + \frac{1}{r+2} disc_N(\hat{P}, \hat{Q}).$$

In addition we have that $disc_N(\hat{P}, \hat{Q}) \leq \sqrt{r}\ MMD(\hat{P}, \hat{Q}) \leq r\ disc(\hat{P}, \hat{Q})$.

Under the uniform distribution, u is unbiased: each direction for the model error is equally likely. This is more optimistic than the assumption of the MMD, where u was biased towards directions that could larger errors on $\hat{P}$. Because now u is not biased, $\mathbb{E}_u G(u, M)$ is smaller under this $p(u)$ than in Theorem 2 and 6 and so this $p(u)$ is more optimistic. The Nuclear Discrepancy (ND) owns its name to the fact that it is proportional to the nuclear matrix norm of M.

An appealing property of this choice of $p(u)$ is that, given a fixed $\hat{P}$, any choice of $\hat{Q}$ does not influence $p(u)$. For the Discrepancy and the MMD, choosing different $\hat{Q}$ leads to different $p(u)$. Thus choosing queries changes the distribution of $p(u)$ and thus also implicitly the distribution of h and f. Instead, for the ND, our queries do not influence the distribution of h and f. This assumption seems reasonable, since f is usually assumed to be fixed and independent of our actions.

Under the uniform distribution the ND provides the tightest bound on the expected loss, while the MMD bound is looser, and the Discrepancy bound is the loosest. Therefore, if this probabilistic assumption is the most accurate, minimization of the Nuclear Discrepancy may lead to the best queries for active learning, followed by the MMD and Discrepancy, in that order[5].

[5] As an aside, note that $\text{MMD}(\hat{P}, \hat{Q}) \leq \text{disc}_N(\hat{P}, \hat{Q})$, since $||\lambda||_2 \leq ||\lambda||_1$. Therefore, by upperbounding the MMD in (3.3) we can also give a (looser) worst-case bound in terms of the ND for the agnostic case.

3.6. EXPERIMENTS

We explain the setup and baselines, afterward we review our main results: the realizable setting. We discuss the results and examine the probabilistic assumptions empirically. Somewhat similar results are observed in the agnostic setting which we will briefly discuss. An additional experiment investigates the influence of subsampling of datasets on our results. This subsampling experiment and all results of the agnostic case are discussed in detail in the Appendix.

3.6.1. EXPERIMENTAL SETUP AND BASELINES

An overview of the experimental procedure is given in Algorithm 1. A training set (65%) and test set (35%) are used — the training set corresponds to $\hat{P}$ and we indicate the testset by $\hat{T}$. We use the active learners to select batches of size $n = 1, 2, \ldots, 50$. For computational reasons we select batches in a sequential greedy fashion. Initially at $t = 0$ the batch is empty: $\hat{Q}_0 = \emptyset$. In iteration $1 \leq t \leq n$ the active learner selects a sample x_t from the unlabeled pool $\hat{U}_{t-1} = \hat{P} \setminus \hat{Q}_{t-1}$ according to $x_t = \operatorname{argmin}_{s \in \hat{U}_{t-1}} \operatorname{obj}(\hat{P}, \hat{Q}_{t-1} \cup s)$. We perform experiments multiple times to ensure significance of the results. We call each repetition a run, and for each run a new training and test split is used. During one run, we evaluate each active learner using the described procedure of Algorithm 1.

Algorithm 1: Zero shot active learning

input : Unlabeled trainingset $\hat{P}$, Testset $\hat{T}$, labeling budget n, active learning criterium $\text{obj} \in \{\text{MMD}, \text{disc}, \text{disc}_N\}$, hyperparameters of model μ, σ

output : MSE performance on testset T

$\hat{Q}_0 \leftarrow \emptyset$; `// Init batch`

$\hat{U}_0 \leftarrow \hat{P}$; `// Init unlabeled pool`

for $t \leftarrow 1$ **to** n **do**

  $x_t \leftarrow \operatorname{argmin}_{s \in \hat{U}_{t-1}} \operatorname{obj}(\hat{P}, \hat{Q}_{t-1} \cup s)$; `// Find optimal query`

  $\hat{Q}_t \leftarrow \hat{Q}_{t-1} \cup x_t$; `// Update batch`

  $\hat{U}_t \leftarrow \hat{P} \setminus \hat{Q}_t$; `// Update unlabeled pool`

end

Request all labels for objects $\hat{Q}_n$ to obtain labeled dataset $\hat{Q}_n^{\text{lab}}$;

Train kernel regularized least squares model h on $\hat{Q}_n^{\text{lab}}$ with hyperparameters μ, σ;

Compute mean squared error (MSE) of h on unseen testset T;

As baseline we use random sampling and a greedy version of the state-of-the-art MMD active learner [8, 14]. We compare the baselines with our novel active learners: the Discrepancy active learner and the Nuclear Discrepancy active learner.

The methods are evaluated on 13 datasets that originate either from the UCI Machine Learning repository [41] or were provided by Cawley and Talbot [42]. See Appendix C.5 for the dataset names and characteristics. Furthermore, we perform an experiment on the image dataset MNIST. The MNIST dataset [43] consists of images of handwritten digits of size 28×28 pixels. By treating each pixel as a feature, the dimensionality of this dataset is

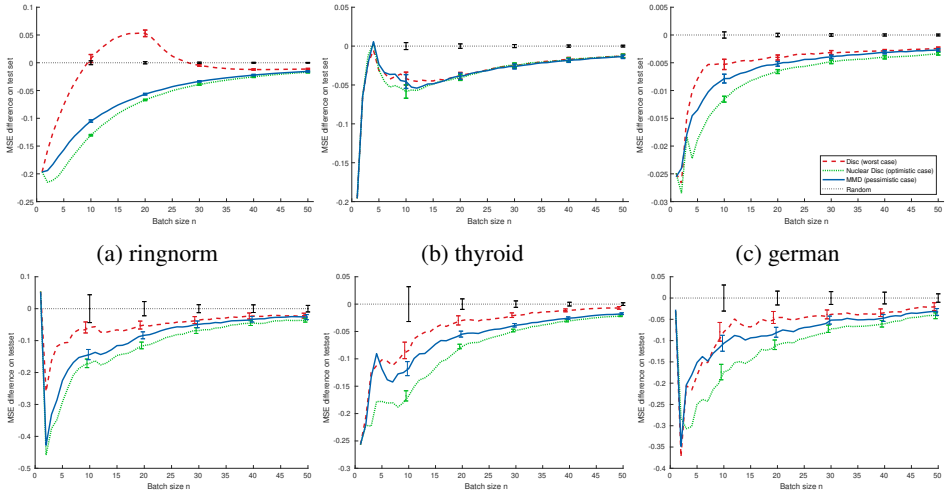
784 which is relatively high dimensional. Like Yang and Loog [23] we construct 3 difficult binary classification problems: 3vs5, 7vs9 and 5vs8.

To make datasets conform to the realizable setting we use the approach of Cortes and Mohri [18]: we fit a model of our hypothesis set to the whole dataset and use its outputs as labels. To set reasonable hyperparameters we use a similar procedure as [5]. We use labeled data before any experiments are performed to perform model selection to determine hyperparameters (σ and μ of the KRLS model). This can be motivated by the fact that in practice a related task or dataset may be available in order to obtain a rough estimate of the hyperparameter settings. This procedure makes sure η_{MMD} and η_{disc} are small in the agnostic setting.

Recall that the active learners minimize bounds on $L_{\hat{P}}(h, f)$. Therefore active learners then implicitly also minimizes a bound on $L_P(h, f)$, see Theorem 1. By choosing hyperparameters in the described way above, we ensure that the Rademacher complexity term $R_m(H)$ is not too large and we do not overfit. We measure performance on an independent test set in order to get an unbiased estimate of $L_P(h, f)$. To aid reproducibility we give all hyperparameters and additional details in Appendix C.5. We set $\sigma_{\mathscr{L}}$ according to our analysis in Corollary 1.

3.6.2. REALIZABLE SETTING

First we benchmark the active learners in the realizable setting. In this setting we are assured that $\eta = 0$ in all bounds and therefore we eliminate unexpected effects that can arise due to model misspecification. We study this scenario to validate our theoretical results and gain more insight, furthermore, note that this scenario is also studied in adaptation [18].



(a) ringnorm (b) thyroid (c) german

(d) mnist 3vs5 (e) mnist 7vs9 (f) mnist 5vs8

Figure 3.1.: Learning curves for several datasets for the realizeable setting. Results are averaged over 100 runs. The MSE is measured with respect to random sampling (lower is better).

Several learning curves are shown in Figure 3.1, all curves can be found in Appendix C.8.1. The MSE of the active learner minus the mean performance (per query) of random sampling is displayed on the y-axis (lower is better). The curve is averaged over 100 runs. Error bars represent the 95% confidence interval of the mean computed using the standard error.

We summarize results on all datasets using the Area Under the (mean squared error) Learning Curve (AULC) in Table 3.2. The AULC is a different metric than the well known AUROC or AUPRC measures. The AUROC measure summarize the performance of a model for different misclassification costs (type I and type II costs) and the AUPRC is useful when one class is more important than the other, such as in object detection.

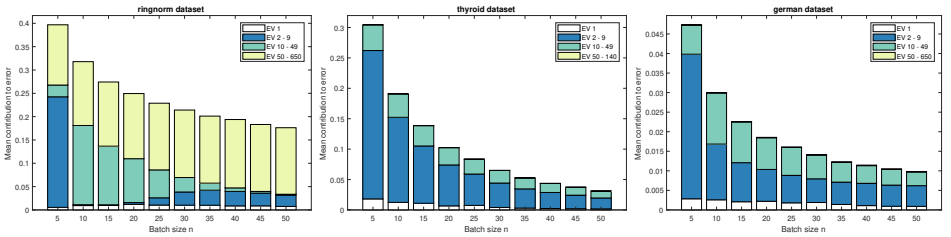
By contrast, AULC is specifically suited to active learning, and summarizes the performance of an active learning algorithm for different number of labeling budgets [44–46]. Low AULC is obtained when an active learner quickly learns a model with low MSE. If a method in the table is bold, it either means it is the best method (as judged by the mean), or if it is not significantly worse than the best method (as judged by the t-test). Significance improvement is judged by a paired two tailed t-test (significance level $p = 0.05$). We may use a paired test since during one run all active learners are evaluated using the same training and test split.

In the majority of the cases the MMD improves upon the Discrepancy (see Table 3.2). The results on the ringnorm dataset are remarkable, here the Discrepancy sometimes performs worse than random sampling, see Figure 3.1. We observe that generally the Discrepancy performs the worst. These results illustrates that tighter worst case bounds do not guarantee improved performance. The proposed ND active learner significantly improves upon the MMD in 9 out of the 13 datasets tested. Here we counted MNIST once, while we remark that on all subproblems the ND improves significantly on the MMD. This provides evidence that the proposed method can also deal with high-dimensional datasets. In case the ND does not perform the best, it ties with the MMD or Discrepancy. The ND never performs significantly worse. This ranking of the methods exactly corresponds to the order of the bounds given by Theorem 7 under our optimistic probabilistic assumptions. This supports our hypothesis that we find ourselves more often in a more optimistic average-case scenario.

Table 3.2.: Area Under the mean squared error Learning Curve (AULC) for the strategies in the realizable setting, averaged over 100 runs. Bold indicates the best result, or results that are not significantly worse than the best result, according to a paired t-test ($p = 0.05$). Parenthesis indicate standard deviation.

Dataset	Random	Discrepancy	MMD	Nuclear Discrepancy
vehicles	11.1 (2.2)	**8.0 (1.0)**	**7.9 (0.9)**	**7.9 (0.9)**
heart	3.5 (0.8)	2.3 (0.3)	2.2 (0.3)	**2.1 (0.3)**
sonar	13.9 (1.7)	12.5 (1.2)	11.9 (1.1)	**11.3 (1.2)**
thyroid	6.8 (1.5)	5.2 (0.9)	**5.1 (0.9)**	**5.0 (1.0)**
ringnorm	13.2 (1.2)	12.7 (0.8)	10.0 (0.3)	**9.4 (0.3)**
ionosphere	7.0 (1.3)	5.6 (0.8)	5.0 (0.8)	**4.6 (0.6)**
diabetes	1.7 (0.4)	**1.2 (0.1)**	**1.2 (0.1)**	**1.2 (0.1)**
twonorm	6.4 (1.2)	4.1 (0.4)	3.7 (0.4)	**3.3 (0.3)**
banana	7.5 (0.9)	5.0 (0.4)	**4.8 (0.3)**	**4.8 (0.3)**
german	1.4 (0.3)	1.2 (0.1)	1.1 (0.1)	**1.0 (0.1)**
splice	10.8 (1.3)	9.9 (0.8)	9.9 (0.9)	**9.0 (0.9)**
breast	3.4 (0.9)	2.1 (0.2)	2.1 (0.2)	**2.0 (0.2)**
mnist 3vs5	29.5 (4.3)	26.9 (2.3)	25.0 (2.1)	**23.8 (1.7)**
mnist 7vs9	13.2 (2.5)	10.9 (1.4)	10.0 (1.0)	**8.9 (0.7)**
mnist 5vs8	30.1 (3.4)	26.9 (2.7)	26.1 (2.3)	**24.5 (2.1)**

3.6.3. DECOMPOSITION OF PROBABILISTIC BOUNDS



(a) ringnorm (b) thyroid (c) german

Figure 3.2.: Decomposition of the sum $G(u, M)$ during active learning for several datasets. EV1 indicates the contribution of λ_1, EV2-9 indicate the summed contributions of $\lambda_2, \ldots, \lambda_9$, etc. Averaged over 100 runs of the random active learner. λ_1 in most cases contributes little and in general all λ_i contribute to $G(u, M)$. This supports the optimistic probabilistic assumptions.

Since we are in the realizable setting we can compute $u = h - f$ with the true labeling function f and our trained model h. Thus we can compute each term in the sum of $G(u, M)$ in (3.7) during the experiments[6]. We show the contribution of each eigenvalue to $G(u, M)$. In Figure 3.2 we show this decomposition using a stacked bar chart during several active learning experiments of the baseline active learner 'Random'[7]. Here EV1 indicates the largest absolute eigenvalue, its contribution is given by $\bar{u}_1^2|\lambda_1|$ (see also (3.7)). EV 2 - 9 to indicate the summed contribution: $\sum_{i=2}^{9} \bar{u}_i^2|\lambda_i|$, etc. The mean contributions over 100 runs are shown.

[6] See Appendix C.4 for details how to compute $G(u, M)$ in case kernels are used.

[7] Results for other strategies are similar. Results on all datasets are given in Appendix C.8.2.

Observe that the contribution of $|\lambda_1|$ to $G(u, M)$ is often small, it is shown by the small white bar at the bottom of the barchart. Therefore the Discrepancy active learner chooses suboptimal samples: its strategy is optimal for a worst-case scenario $G(u, M) = 4\Lambda^2|\lambda_1|$ that is very rare. We observe that typically all λ_i contribute to $G(u, M)$ supporting our probabilistic assumption.

3.6.4. AGNOSTIC SETTING

For completeness, we briefly mention the agnostic setting, for all details see Appendix C.6. In the agnostic setting the rankings of methods can change and performance differences become less significant. The ND still improves more upon the MMD than the reverse, however, the trend is less significant. Because our assumption $\eta = 0$ is violated our theoretical analysis is less applicable.

For the MNIST experiments we however find that the results for some subproblems almost coincides with the realizeable setting: apparently, for the MNIST dataset the model misspefication is very small. This may be because the dataset is of relatively high dimensionalion.

3.6.5. INFLUENCE OF SUBSAMPLING

We briefly mention an additional experiment that we have performed on the splice dataset to see how subsampling affects performance. To this end we measure the performance while we vary the pool size $\hat{P}$ by changing the amount of subsampling. This to investigate how the proposed methods would perform for problems with a larger scale. For all details see Appendix C.7, here we will be brief.

For small pool sizes all active learners experience a drop in performance. We find the larger the pool, the better the performance, up until some point at which the performance levels off. The experiment provides evidence that if finer subsampling is used or larger datasets are used, methods typically improve in performance up to a point where performance levels off.

3.7. DISCUSSION

In the experiments we have observed that in the realizable setting the order of the bounds under our more optimistic probabilistic assumptions give the best indication of active learning performance. The empirical decomposition of $G(u, M)$ during experiments also supports our hypothesis that we generally find ourselves in a more optimistic scenario instead of a worst case scenario.

Still it is meaningful to look at worst-case guarantees, though the worst-case should be expected to occur. The worst-case assumed by the Discrepancy can never occur in the realizable setting, and we believe it is also highly unlikely in the agnostic case. The strength of our approach is that it considers all scenarios equally and does not focus too much on specific scenarios, making the strategy more robust.

Our work illustrates that the order of bounds can change under varying conditions and thus tightness of bounds is not the whole story. The conditions under which the bounds hold are equally important, and should reflect the mathematical setting as much as possible.

For example, in a different setting where an adversary would pick u, the Discrepancy active learner would be most appropriate. This insight illustrates that not only by obtaining tighter bounds active learning performance can be improved, but by finding more appropriate assumptions (bound-based) active learners can be improved as well.

Our work supports the idea of Germain *et al.* [36] who introduce a probabilistic version of the Discrepancy bound for the zero-one loss [47]. Our conclusions also support that the direct Cortes *et al.* [37] takes: by using more accurate assumptions to better characterize the the worst case scenario, performance may be improved.

In our study we have focused on minimizing the mean squared error. It would be interesting to investigate the extension of the Nuclear Discrepancy to other loss functions, in particular the zero-one loss. As far as we can see, however, such an extension is not trivial. The above mentioned probabilistic version of the Discrepancy by Germain *et al.* [36] may provide some inspiration to achieve this, but they offer a PAC Bayes approach that cannot be easily adapted to our setting.

Where the experiments in the realizable setting provide clear insights, the results concerning the agnostic setting are not fully understood. A more in depth experimental study of the agnostic setting is complicated by unexpected effects of η. Since probabilistic bounds are the most informative in the realizable setting, it is of interest to consider probabilistic bounds for the agnostic setting as well.

In our experiments we have used greedy optimization to compute the batch $\hat{Q}_n$. It is theoretically possible to optimize a whole batch of queries in one global optimization step. However, for the MMD this problem is known to be NP-hard [14]. Minimizing the Discrepancy is also non-trivial, as illustrated by the involved optimization procedure required by Cortes and Mohri [18] for domain adaptation. Note that their optimization problem is easier than the optimization problem posed by active learning, where binary constraints are necessary. Since the objective value of the Nuclear Discrepancy is given by an expectation which can be approximated using sampling, we believe further speed ups are possible.

In this work we have only considered single-shot batch active learning. In regular batch-mode active learning label information of previously selected samples can be used to improve query selection. This can be accommodated in our active learner by refining $p(u)$ using label information. Our results have implications for adaptation as well. We suspect our suggested choice of $\sigma_{\mathscr{L}}$ may improve the MMD domain adaptation method [17]. Furthermore, our results suggest that the ND is a promising objective for adaptation.

3.8. CONCLUSION

To investigate the relation between generalization bounds and active learning performance, we gave several theoretical results concerning the bound of the MMD active learner and the Discrepancy bound. In particular, we showed that the Discrepancy provides the tightest worst-case bound. We introduced a novel quantity; Nuclear Discrepancy, motivated from optimistic probabilistic assumptions derived from the principle of maximum entropy. Under these probabilistic assumptions the ND provides the tightest bound on the expected loss, followed by the MMD, and the Discrepancy provides the loosest bound.

Experimentally, we observed that in the realizable setting the Discrepancy performs the worst, illustrating that tighter worst-case bounds do not guarantee improved active learning

performance. Our optimistic probabilistic analysis clearly matches the observed behavior in the realizable setting: the proposed ND active learner improves upon the MMD, and the MMD improves upon the Discrepancy active learner. We find that even on the high-dimensional image dataset MNIST our method is competitive. A similar, weaker, trend is observed in the agnostic case. One of our key conclusions is that not only bound tightness is important for active learning performance, but that appropriate assumptions matter equally so.

3.9. BIBLIOGRAPHY

[1] T. J. Viering, J. H. Krijthe, and M. Loog, *Nuclear discrepancy for single-shot batch active learning,* Machine Learning **108**, 1561 (2019).

[2] B. Settles, *Active Learning,* Synthesis Lectures on Artificial Intelligence and Machine Learning **6**, 1 (2012).

[3] D. Cohn, L. Atlas, and R. Ladner, *Improving generalization with active learning,* Machine Learning **15**, 201 (1994).

[4] Q. Gu and J. Han, *Towards Active Learning on Graphs: An Error Bound Minimization Approach,* in *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM)* (2012) pp. 882–887.

[5] Q. Gu, T. Zhang, J. Han, and C. H. Ding, *Selective Labeling via Error Bound Minimization,* in *Proceedings of the 25th Conference on Advances in Neural Information Processing Systems (NIPS)* (2012) pp. 323–331.

[6] R. Ganti and A. Gray, *UPAL: Unbiased Pool Based Active Learning,* in *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)* (2012) pp. 422–431.

[7] Q. Gu, T. Zhang, and J. Han, *Batch-Mode Active Learning via Error Bound Minimization,* in *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI)* (2014).

[8] Z. Wang and J. Ye, *Querying Discriminative and Representative Samples for Batch Mode Active Learning,* in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (2013) pp. 158–166.

[9] R. Rifkin, G. Yeo, and T. Poggio, *Regularized least-squares classification,* Advances in Learning Theory: Methods, Model, and Applications **190**, 131 (2003).

[10] S.-j. Huang, R. Jin, and Z.-h. Zhou, *Active Learning by Querying Informative and Representative Examples,* in *Proceedings of the 23th Conference on Advances in Neural Information Processing Systems (NIPS)* (2010) pp. 892–900.

[11] G. Contardo, L. Denoyer, and T. Artieres, *A meta-learning approach to one-step active learning,* arXiv preprint arXiv:1706.08334 (2017).

[12] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, *Dermatologist-level classification of skin cancer with deep neural networks,* Nature **542**, 115 (2017).

[13] A. S. Harpale and Y. Yang, *Personalized active learning for collaborative filtering,* in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (ACM, 2008) pp. 91–98.

[14] R. Chattopadhyay, Z. Wang, W. Fan, I. Davidson, S. Panchanathan, and J. Ye, *Batch Mode Active Sampling Based on Marginal Probability Distribution Matching,* in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (2012) pp. 741–749.

[15] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, *A Kernel Two-sample Test,* Machine Learning Research **13**, 723 (2012).

[16] Y. Mansour, M. Mohri, and A. Rostamizadeh, *Domain Adaptation: Learning Bounds and Algorithms,* in *Proceedings of the 22nd Annual Conference on Learning Theory (COLT)* (2009).

[17] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf, *Correcting sample selection bias by unlabeled data,* in *Proceedings of the 19th Conference on Advances in Neural Information Processing Systems (NIPS)* (2007) pp. 601–608.

[18] C. Cortes and M. Mohri, *Domain adaptation and sample bias correction theory and algorithm for regression,* Theoretical Computer Science **519**, 103 (2014).

[19] C. Tosh and S. Dasgupta, *Diameter-based active learning,* arXiv preprint arXiv:1702.08553 (2017).

[20] B. Settles, *From theories to queries: Active learning in practice,* in *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010* (2011) pp. 1–18.

[21] J. Attenberg and F. Provost, *Inactive learning?: difficulties employing active learning in practice,* ACM SIGKDD Explorations Newsletter **12**, 36 (2011).

[22] M. Loog and Y. Yang, *An empirical investigation into the inconsistency of sequential active learning,* in *2016 23rd International Conference on Pattern Recognition (ICPR)* (IEEE, 2016) pp. 210–215.

[23] Y. Yang and M. Loog, *A benchmark and comparison of active learning for logistic regression,* Pattern Recognition **83**, 401 (2018).

[24] Z. Xu, K. Yu, V. Tresp, X. Xu, and J. Wang, *Representative Sampling for Text Classification Using Support Vector Machines,* in *Advances in Information Retrieval* (Springer, 2003) pp. 393–407.

[25] K. Yu, J. Bi, and V. Tresp, *Active Learning via Transductive Experimental Design,* in *Proceedings of the 23rd International Conference on Machine Learning (ICML)* (2006) pp. 1081–1088.

[26] S. C. Hoi, R. Jin, J. Zhu, and M. R. Lyu, *Batch mode active learning and its application to medical image classification,* in *Proceedings of the 23rd International Conference on Machine Learning (ICML)* (2006) pp. 417–424.

[27] M.-F. Balcan, A. Beygelzimer, and J. Langford, *Agnostic active learning,* Journal of Computer and System Sciences **75**, 78 (2009).

[28] S. Hanneke, *A bound on the label complexity of agnostic active learning,* in *Proceedings of the 24th International Conference on Machine learning (ICML)* (2007) pp. 353–360.

[29] M.-F. Balcan and R. Urner, *Active learning–modern learning theory,* Encyclopedia of Algorithms , 8 (2016).

[30] Z. BodÃ, Z. Minier, and L. CsatÃ, *Active learning with clustering,* in *Active Learning and Experimental Design workshop in conjunction with AISTATS 2010* (2011) pp. 127–139.

[31] R. Hu, B. Mac Namee, and S. J. Delany, *Off to a good start: Using clustering to select the initial training set in active learning.* in *FLAIRS Conference* (2010).

[32] J. Zhu, H. Wang, T. Yao, and B. K. Tsou, *Active learning with sampling by uncertainty and density for word sense disambiguation and text classification,* in *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1* (2008) pp. 1137–1144.

[33] H. T. Nguyen and A. Smeulders, *Active learning using pre-clustering,* in *Proceedings of the 21rst International Conference on Machine learning (ICML)* (2004) p. 79.

[34] R. Urner, S. Wulff, and S. Ben-David, *Plal: Cluster-based active learning,* in *Conference on Learning Theory (COLT)* (2013) pp. 376–397.

[35] R. Chattopadhyay, W. Fan, I. Davidson, S. Panchanathan, and J. Ye, *Joint Transfer and Batch-mode Active Learning,* Proceedings of the 30th International Conference on Machine Learning (ICML) , 253 (2013).

[36] P. Germain, A. Habrard, F. Laviolette, and E. Morvant, *A pac-bayesian approach for domain adaptation with specialization to linear classifiers,* in *Proceedings of the 30th International Conference on Machine Learning (ICML)* (2013) pp. 738–746.

[37] C. Cortes, M. Mohri, and A. M. Medina, *Adaptation based on generalized discrepancy,* Journal of Machine Learning Research **20**, 1 (2019).

[38] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning* (MIT press, Cambridge, Massachusetts, 2012).

[39] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis* (Cambridge University Press, Cambridge, UK, 2004).

[40] E. T. Jaynes, *Information theory and statistical mechanics,* Physical review **106**, 620 (1957).

[41] M. Lichman, *UCI Machine Learning Repository,* (2013).

[42] G. C. Cawley and N. L. Talbot, *Fast exact leave-one-out cross-validation of sparse least-squares support vector machines,* Neural Networks **17**, 1467 (2004).

[43] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, *Gradient-based learning applied to document recognition,* Proceedings of the IEEE **86**, 2278 (1998).

[44] J. ONeill, S. J. Delany, and B. MacNamee, *Model-free and model-based active learning for regression,* in *Advances in Computational Intelligence Systems* (Springer, 2017) pp. 375–386.

[45] M. Huijser and J. C. van Gemert, *Active decision boundary annotation with deep generative models,* in *Proceedings of the IEEE International Conference on Computer Vision* (2017) pp. 5286–5295.

[46] B. Settles and M. Craven, *An analysis of active learning strategies for sequence labeling tasks,* in *Proceedings of the conference on empirical methods in natural language processing* (Association for Computational Linguistics, 2008) pp. 1070–1079.

[47] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, *A theory of learning from different domains,* Machine Learning **79**, 151 (2010).

4

THE SHAPE OF LEARNING CURVES: A REVIEW

Learning curves provide insight into the dependence of a learners generalization performance on the training set size. This important tool can be used for model selection, to predict the effect of more training data, and to reduce the computational complexity of model training and hyperparameter tuning. This review provides a formal definition of the learning curve, and briefly covers basics such as its estimation. Our main contribution is a comprehensive overview of the literature regarding the shape of learning curves. We discuss empirical and theoretical evidence in favor of the shape of a power law or an exponential. We draw specific attention to examples of learning curves that are ill-behaved, showing worse learning performance with more training data. To wrap up, we point out various open problems that warrant deeper empirical and theoretical investigation. All in all, our review underscores that learning curves are surprisingly diverse and no universal model can be identified.

This work was accepted at IEEE Transactions on Pattern Analysis and Machine Intelligence and a preprint is available at [1]. This chapter is a more detailed version than the TPAMI or arxiv versions, but does not cover Gaussian Process learning curves.

4.1. INTRODUCTION

A *learning curve* is an important, graphical representation that can provide insight into such learning behavior by plotting the expected generalization performance against the number of training examples used for learning.

We review learning curves in the context of standard supervised learning problems such as classification and regression. The primary focus is on the shapes that learning curves can take on. We make a distinction between well-behaved learning curves that show improved performance with more data and ill-behaved learning curves that, perhaps surprisingly, do not. We discuss theoretical and empirical evidence in favor of different shapes, underlying assumptions made, how knowledge about those shapes can be exploited, and further results of interest. In addition, we provide the necessary background to interpret and use learning curves as well as a comprehensive overview of the important research directions.

4.1.1. OUTLINE

The next section starts off with a definition of learning curves and discusses how to estimate them in practice. We also give several recommendations on how to plot, summarize and fit learning curves. It also briefly considers what we call Problem-Average (PA) learning curves, which are primarily used in Bayesian settings, and *feature curves*, which offer a complementary view to the learning curve.

Section 4.3 covers the most important applications of learning curves, such as the insight into model selection they can give us, and how they are employed in meta-learning, and how they can be used to reduce the cost of labeling or computation. For these tasks, it is essential that we know what kind of shape we can expect (e.g. exponential, power law, etc.). Therefore, the rest of this survey is concerned with the question: what kind of shapes do learning curves have in typical machine learning settings according to the literature (empirical and theoretical studies)? And what influences the shape?

Section 4.4 considers empirical evidence supporting well-behaved learning curves: curves that generally show improved performance with more training data. Learners that satisfy this property are called smart [2, page 106] and monotone [3] (Chapter A). We review the parametric models that have been considered for modeling learning curves, and we find that several are redundant, and uncover many aspects of curve fitting that are often overlooked. For shallow learners (such as decision trees, logistic regression, etc.) we find that most empirical studies that investigate the shape contradict each other. Recently, two large scale studies try to improve matters, and find that learning curves are best modeled by a power laws with 3 parameters or MMF4 or WBL4, two specific parametric models that have 4 parameters. Interestingly, parametric models with 4 parameters were discounted in earlier studies to avoid overfitting learning curve data, but in some cases their extra flexibility offered by an additional parameter does seem useful for learning curve fitting. Overall, there is some evidence that no curve model seems to be systematically better than the other. In the context of deep learning, empirical evidence in favor of the power law is much more convincing, but requires careful and costly tuning of hyperparameters, otherwise the power law disappears and learning becomes slower.

Section 4.5 reviews some of the theoretical works that support the power law and exponential shape of learning curves. We discuss what factors and assumptions influence the

shape, such as the complexity of the hypothesis class, how separable a classification problem is, and whether assumptions of machine learning models are correct. We also discuss the Problem-Average learning curves and their shapes, and when these show guaranteed improvement with more data. We discuss that many theoretical works (such as those relying on the notion of PAC-learning) do not tell us much about the shape of learning curves.

Section 4.6 then follows with an overview of important examples of learning curves that do not behave well. These are not an artefact of the experimental setup or due to unlucky sampling; in fact we discuss the rigorous theoretical proofs behind these results. Examples are learning curves that display multiple local maxima, curves with local minima, or curves that display large jumps. We discuss possible causes and ways to mitigate these behaviors, with at the end of the section a possible universal solutions to make learning curves well-behaved for any learner. We believe that especially this section shows that our understanding of the behavior of learners is more limited than one might expect.

Section 4.7 provides an extensive discussion. We give several concrete open questions and offer plenty of avenues for follow-up work on learning curves. Then Section 4.8 reiterates the most important points of this chapter and concludes it. The remainder of the current section discusses the meanings of the term "learning curve" and its synonyms in the machine learning literature.

4.1.2. LEARNING CURVE TERMINOLOGY

To avoid any confusion we will use the terms training and learning curve. It should be noted that previous work defined many different names that relate to our definition. We use training curve exclusively to refer to the curves that visualize performance during training. For example, a plot of the training error or loss (or any other optimization objective) versus the number of epochs, is an example of a training curve. Especially in the neural network literature, this is often called a learning curve [4, 5], but we denote this as a training curve.

The learning curve visualizes the performance of the learning algorithm versus the size of the training set. We will make this definition more rigorous in the next section. There are also various synonyms in literature for learning curve, such as *error curve*, *experience curve*, *improvement curve* and *generalization curve* [4, 6, 7].

4.2. DEFINITION, ESTIMATION, FEATURE CURVES

This section makes the notion of a learning curve more precise and describes how learning curves can be estimated from data. We give some recommendations when it comes to plotting, summarizing and fitting learning curves. In particular, we highlight several pitfalls that are typically made in works that fit learning curves. Finally, feature curves offer a view on learners that is complementary to that of learning curves. These and combined learning-feature curves are covered at the end of this section.

4.2.1. DEFINITION OF LEARNING CURVES IN LEARNING THEORY

Here we define the theoretical concept of a learning curve, in the next subsection we discuss how to estimate it in practice. Let S_n indicate a training set of size n. In standard classification and regression, S_n consists of (x, y) pairs, where $x \in \mathcal{X}$, where often $\mathcal{X} \subseteq \mathbb{R}^d$ is the

d-dimensional input vector (i.e., the features, measurements, or covariates) and $y \in \mathcal{Y}$ is the corresponding output. For binary classification, $\mathcal{Y} \subseteq \{-1, +1\}$ and for regression $\mathcal{Y} \subseteq \mathbb{R}$. The (x, y) pairs of the training set are i.i.d. samples of an unknown probability distribution P_{XY} over $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. This is a standard assumption in learning theory. Here P_{XY} completely determines one particular *learning problem*.

We denote a learner as A, which takes as input the samples S_n and outputs a predictor or hypothesis h. A predictor h is an element of a prespecified hypothesis class H. H contains all models that can be returned by the learner A. An example of a hypothesis class is the set of all linear models $\{h : x \mapsto a^T x + b | a \in \mathbb{R}^d, b \in \mathbb{R}\}$. Examples of typical learning algorithms for A are empirical risk minimizers (ERMs) or the 1 nearest neighbour classifier. More formally, A is a particular mapping from the set of all sample sets $\mathcal{S} := \mathcal{Z} \cup \mathcal{Z}^2 \cup \mathcal{Z}^3 \cup \ldots$ to elements of the hypothesis class H. That is, $A : \mathcal{S} \to H$.

Note that we do not consider arbitrary algorithms A as their learning curve may behave arbitrarily. To see that, let $h_1, h_2, \ldots$, be an ordering of all hypotheses that is determined before observing any data. Let A map each S_n to h_n, or in other words, the returned hypotheses only depends on the size of S_n. For such algorithms A any curve can be achieved by reordering the hypotheses, and in fact, the samples (x, y) are not used for learning.

Instead, we consider typical learning algorithms used in practice, such as decision trees, empirical risk minimization, etc. These learning curves for machine learning algorithms will (hopefully) have more regularity in them then the algorithm above. However, we are not aware of any formal characterization of all learning algorithms (that excludes the counter example above), and we believe this is an interesting open problem. It is however out of the scope of this chapter.

For $x \in \mathcal{X}$, h provides a prediction $h(x) \in \mathcal{Y}$, which we denote as $\hat{y}$. The performance of a particular hypothesis h is measured by a loss function l that compares y to $\hat{y}$. For regression the squared loss can be used: $l_{\text{sq}}(y, \hat{y}) = (y - \hat{y})^2$. For binary classification the zero-one loss can be used: $l_{01}(y, \hat{y}) = \frac{1}{2}(1 - y\hat{y})$ since $\mathcal{Y} \subseteq \{-1, +1\}$.

The typical goal is that a predictor performs well on average on new and unseen observations. Ideally, this is measured by the expected loss or risk R over the true distribution P_{XY}:

$$R(h) = \int L(y, h(x)) dP(x, y). \tag{4.1}$$

For the zero-one loss, the risk corresponds to the error rate of the classifier h, in other words, the probability of making a mistake on new data. The risk cannot be calculated directly since the distribution P_{XY} is unknown in practice and thus the risk needs to be estimated. However, in simulation studies where a particular P_{XY} is assumed, it is possible to compute this quantity. In most that follows, we omit the subscript XY for P_{XY}.

Now, an *individual* learning curve considers training sets S_n for $n \in \mathbb{N}$ and calculates the corresponding risks $R(A(S_n))$ as a function of n. However, a single S_n may deviate significantly from the expected behavior. Therefore, we are often interested in an average over many different sets S_n, and ideally the expectation

$$\bar{R}_n(A) = \mathop{\mathbb{E}}_{S_n \sim P^n} R(A(S_n)). \tag{4.2}$$

The plot of $\bar{R}_n(A)$ against the training set size n gives us the (expected) learning curve. From this point onward, when we talk about *the* learning curve, this is what is meant. Note

that, we *can* consider monotonic training sets, meaning that $S_{n-1} \subset S_n$ (thus that additional samples are always appended) or non-monotonic training sets where S_{n-1} and S_n are independently sampled from P. In some theoretical analyses monotonic training sets are required; otherwise, which is desired depends on the application. Since this issue is closely related to learning curve estimation, we postpone this discussion to the next subsection.

For classification, often one generates a learning curve using a stratified training set. In that case, if we have k classes, the training set consists of $n = kn'$ training samples, where n' samples are taken per class. In that case, the learning curve can plot $\bar{R}_{n'}(A)$ against n'. Note that this typically reduces the variance of the learning curve, since some learners may become unstable if the training set is unbalanced. It depends on the setting whether samples can be collected per class.

The preceding learning curve is defined for a single distribution P. Sometimes we wish to study how a model performs over a range of problems or, more generally, a distribution $\mathscr{P}$ over problems. The learning curve that considers such averaged performance is referred to as the problem-average (PA) learning curve:

$$\bar{R}_n^{\text{PA}}(A) = \mathop{\mathbb{E}}_{P \sim \mathscr{P}} \bar{R}_n(A). \tag{4.3}$$

The general term problem-average was coined in [8]. PA learning curves make sense for Bayesian approaches in particular, where an assumed prior over possible problems often arises naturally. The risk integrated over the prior, in the Bayesian literature, is also called the Bayes risk, integrated risk, or preposterior risk [9, page 195]. The term preposterior signifies that, in principle, we can determine this quantity without observing any data. Note that, the PA curve which averages over problems may mask the behavior of individual learning curves for a single problem, an issue that we discuss in more detail in Section 4.7.3.

Let us discuss Bayesian linear regression to illustrate the PA learning curve. Assume there is a distribution P_X over $\mathscr{X}$. Let $y = w^T x + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$ is i.i.d. Gaussian noise with zero mean and variance σ^2 (referred to as Gaussian likelihood). w is the parameter of the true (but unknown for the learner) regression model. For a fixed w, this defines a single problem P_{XY}. To define the problem-average $\mathscr{P}$, we can consider a prior distribution P_W over w, such as a Gaussian with zero mean and identity covariance matrix. To generate a problem-average learning curve, one first samples w from the prior P_W. For each w, where each w corresponds to a particular problem, we generate datasets S_n by sampling x from P_X and sampling y from $N(w^T x, \sigma^2)$. This data can be used to train a learner and to estimate the learning curve for each problem. We repeat this for multiple w, and average the resulting learning curves over problems to approximate the problem-average learning curve of Equation 4.3.

In semi-supervised learning [10] and active learning [11], it can be of additional interest to study the learning behavior as a function of the number of *unlabeled* and *actively selected* samples, respectively.

4.2.2. ESTIMATING LEARNING CURVES IN PRACTICE

In practice, we merely have a finite sample from P and we cannot measure $R(h)$ or consider all possible training sets sampled from P. Often, we can only get an estimate of the learning

curve. Popular approaches are to use a hold-out dataset or k-fold cross validation for this [12–15] as also apparent from the Weka documentation [16] and Scikit-learn implementation [17]. Using cross validation, k folds, random subsets of the dataset, are generated. Each subset serves once as testset, and for each such testset the other subsets serve as the training set. The size of the training set S_n is varied over a range of values by removing samples from the originally formed training set. For each size the learning algorithm is trained and the performance is measured on the test fold. The process is repeated for all k folds, leading to k individual learning curves. The final estimate is their average. The variance of the estimated curve can be reduced by carrying out the k-fold cross validation multiple times [18] and this technique is used in various works that apply learning curves [12–14, 19]. Stratified cross validation may be used to further reduce the variance in case of a classification setting.

Using cross validation to estimate the learning curve has some drawbacks. For one, when making the training set smaller, not using the discarded samples for testing seems wasteful. Also note that the training fold size limits the range of n of the estimated learning curve, especially if k is small. Directly taking a random training set of the preferred size and leaving the remainder as a test set can be a good alternative. This can be repeated to come to an averaged learning curve. This recipe—employed, for instance in the machine learning toolbox PRTools [20], but also by Hess and Wei [19]—allows for easily using any n, leaving no sample unused. Note that the test risks are not independent for this approach, in contrast to cross-validation, since the test sets will have overlap as samples are not discarded. Furthermore, one should realize that for large n, the testset could become small, leading to a larger variance of the estimated risk. Alternatively, the bootstrap (sampling with replacement) can also be considered to come to variable sized training sets [21, 22].

One essential choice is whether one should use monotonic training sets for estimating the learning curve. Currently it is unclear what yields the best estimate for the learning curve. However, if one intends to use the learning curve in a setting where data is scarce or expensive, it makes more sense to use monotonic training sets to cope with the data scarcity, because sampling a training set from scratch is wasteful. In case little data is available and non-monotonic training sets are necessary, one can resort to bootstrapping to simulate the procedure. If data is plenty, one can easily sample each training set independently. In some cases, learners can be updated with new samples (incremental learning), in that case monotonic training sets may also offer computational advantages. Finally, one should take into consideration how the learning curves will be applied. This, we would argue, is the most important in determining whether training sets should be monotonically increasing. In particular, if a learning curve is used to determine the value of gathering additional data, a setting in which data is usually scarce, one should most likely use monotonically increasing training sets or simulate them using bootstrapping.

An altogether different way to learning curve estimation is to assume an underlying parametric model for the learning curve and fit this to the learning curves estimates obtained via approaches described previously. The approach is not widespread under practitioners, but is largely confined to the research work that studies and exploits the general shape of learning curves (see Subsection 4.4.2).

Finally, note that all of the foregoing pertains to PA learning curves as well. In that setting, we may occasionally be able to exploit the special structure of the assumed problem prior.

This is the case, for instance, with Gaussian process regression (which we do not cover in this chapter in detail), where problem averages can sometimes be computed with little cost.

4.2.3. PLOTTING RECOMMENDATIONS

When plotting the learning curve, it can be useful to consider logarithmic axes. Plotting n linearly may mask small but non-trivial gains [23]. Also from a computational standpoint it often makes sense to have n traverse a logarithmic scale [24] (see also Subsection 4.3.2). A log-log or semi-log plot can be useful if we expect the learning curve to display power-law or exponential behavior (Section 4.4), as the curve becomes straight in that case.

For example, let the risk be modeled by $y(n) = ab^n$. In that case, $\log(y(n)) = \log(a) + n\log(b)$, and thus transforming $y(n)$ to log-scale makes the learning curve linear in n (straight) if the learning curve really is of the form ab^n. The same holds for power-law models, where both $y(n)$ and n need to be transformed to log-scale. Note that this only works if there is a non-zero offset to the power-law or exponential. If we are working with the model $y(n) = ab^n + c$, first the offset offset c should be subtracted from y before the log transformation, then after the log transformation the learning curve becomes linear if it truly has the form $ab^n + c$.

In such a plot, it can be easier to discern small deviation from exponential or power law behavior, because we have to discern whether the line is straight or not. Finally, it is common to use error bars to indicate the variability of the learning curve. One may either use the standard deviation over the folds or standard error.

4.2.4. RECOMMENDATIONS FOR SUMMARIZING LEARNING CURVES

It may be useful at times to summarize learning curves into a single number, especially when faced with pagelimits of scientific publications. A popular metric to that end is the area under the learning curve (AULC) [25–27]. To compute this metric, one needs to settle at a number of sample sizes. One then averages the performance at all those sample sizes to get (an approximation of) the area under the learning curve. The AULC thus makes

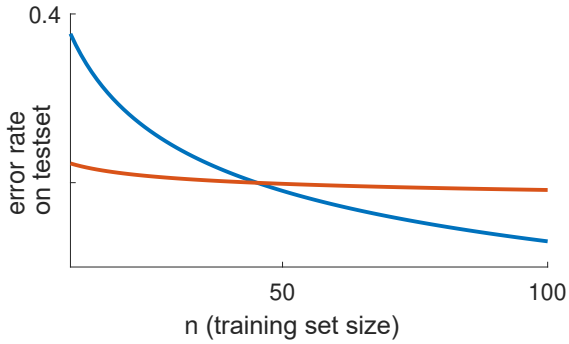


Figure 4.1.: Crossing learning curves. Red starts at a lower error, while blue reaches a lower error rate given enough data. The AULC is approximately equal for both curves.

the curious assumption that all sample sizes should contribute equally to the numerical summary of the learning curve. Furthermore, determining the sample sizes may have a huge influence on the numerical summary. For example, one may wonder whether these sample sizes should be on a logarithmic or linear scale.

Important information can get lost when summarizing. The measure is, for instance, not able to distinguish between two methods whose learning curves cross (Figure 4.1), i.e., where the one method is better in the small sample regime, while the other is better in the large sample setting. Others have proposed to summarize learning curves with the asymptotic value of the learning curve and the number of samples to reach it [28], or the exponent of a power-law fit [29].

These summaries are likely inadequate. The asymptotic value may tell us little about the actual performance for finite sample sizes, and one may wonder when the asymptotic value is reached or how to accurately estimate it. The exponent of the power-law fit may be a logical choice to describe limiting behavior in view of big O notation. However, such a summary hides multiplicative constants and the asymptotic value of the risk. Therefore this summary is inadequate to extrapolate curves or to determine absolute performances. Furthermore, big O notation considers the behavior of the curve in the limit of large n. It is questionable whether a learning curve when extrapolated so far will still behave in that manner in this regime. Therefore we recommend big O notation only to describe relative performances of a learner compared to itself (e.g. to check convergence) for a very coarse summary.

Recently, Hoiem *et al.* [30] suggested to summarize curves using *all* fit parameters (not *only* the exponent). We would recommend this approach, however, one should try multiple parametric models and report the parameters of the best fit *and* the fit quality. In cases when the fit quality is insufficient, one could instead visualize the curve for further investigation, as this can be an indication of an ill-behaved learning curve. However, a reliable way to detect ill-behaving learning curves remains an open problem.

4.2.5. CURVE FITTING RECOMMENDATIONS

Popular parametric forms for fitting learning curves are given in Table 4.1 and their performance is discussed in Section 4.4.2. Here, we make a few recommendations how to properly fit and study empirical learning curves. We also mention some common pitfalls in setting up curve fitting experiments. Some of the paragraphs describing the pitfalls of this section are named explicitly (pitfall F,B,E and S), so that we can refer to them when we discuss the curve fitting studies in Section 4.4.2.

Pitfall F: not Fitting using a non-linear toolbox. Some works [31–36] seem to perform simple least squares fitting on log-values in order to the fit power law POW2 or exponential EXP2 (see Table 4.1). In other words, instead of fitting R_n (as y-values) and n (x-values), they fit $\log(\bar{R}_n)$ and n (for EXP2), and $\log(\bar{R}_n)$ and $\log(n)$ (for POW2). For older works [31, 35] this can be excused, as curve fitting toolboxes were perhaps not so widespread. However, nowadays, one should resort to nonlinear curve fitting techniques. Since, if one would like to find the optimal parameters in the original space (not in the log-space) in terms of the mean squared error, non-linear curve fitting techniques are necessary (for example using LevenbergMarquardt [36], Gauss-Newton, Newton's method, etc.). This is necessary because the mean squared error in the log-space is essentially different than the mean

squared error in the original space. To emphasize the importance, Singh [33] perform fits in the log-space and doubt the validity of some of their results due to this reason.

Pitfall B: not including a Bias term in a parametric model. Fitting log-values only works for POW2 and EXP2, not for models that include an offset or bias term such as EXP3 and POW3. Because for many problems finding a predictor with zero risk is impossible or very difficult, the offset will often be crucial for a good fit. Therefore, we recommend to always include a bias term in a parametric model. Otherwise, for example, a curve may actually have an exponential shape, but the fit for EXP2 may not work at all because no bias term was included.

While we do not name an explicit pitfall in this paragraph, we would like to stress to include sufficient information for reproducibility. We recommend to give the name of the fitting procedure, its parameters (e.g. number of iterations and stopping condition), and indicate how initial points were generated. Sometimes fitting procedures report a failure; in this case it should be clearly mentioned what is done with such results. Some studies use a geometric schedule for the training set sizes (instead of a linear one) to speed up experiments, this should also be mentioned clearly as this may also impact fitting studies.

Pitfall E: Extrapolation (or interpolation) should be done to unseen learning curve data. For all goals involving learning curves, we want to interpolate or extrapolate the learning curve to previously *unseen* training set sizes. This is a generalization task and we have to deal with the problems that this may entail, such as overfitting to learning curve data which was observed by Gu *et al.* [36] and Kolachina *et al.* [37] empirically. Thus, learning curve data should also be split in train and test sets for a fair evaluation. This means that, some points along the learning curve are used for fitting curve models, while other points not seen during fitting are used for evaluating the parametric model.

Typically, extrapolation is done to training set sizes larger than those used for fitting, because we want to know the value of gathering more data. It is standard to evaluate on all those fitting points, the next unseen point, or to evaluate on the largest training set size available. For interpolation, it is common to report the mean squared error on the points used for fitting. Most studies until now evaluate in terms of the mean squared error. Some studies report the well-known R^2 value of the fit; this metric is always determined on the points used for fitting, while one is usually more interested in extrapolation for learning curves. It has been shown that the R^2 is not adequate for nonlinear curve fitting evaluation [38]. As such we suggest to avoid it altogether when evaluating learning curve fitting.

Pitfall S: missing of statistical Significance of the results. We recommend to perform a statistical analyses to determine significance of results of curve fitting. We recommend to perform either a rank-test on the performances, as the performance on test data may not be normally distributed (invalidating assumptions of the t-test) [39, 40]. Repeated measurements can be generated by multiple randomized initialisations of the fitting procedure, by fitting different individual learning curves (obtained by different train/test sets), or by considering different learners or datasets. The appropriate choice is not trivial and depends on the goal of the study and should be carefully considered.

4.2.6. PSEUDO-FISHER'S LINEAR DISCRIMINANT (PSEUDO-FISHER)

We will multiple times refer to a classification model called Pseudo-Fisher's Linear Discriminant (Pseudo-Fisher). Because this model has a closed form solution, it is more amendable

to theoretical analyses, which explains why it is often studied. This classifier is one of the simplest classifiers; for a binary classification problem, it encodes $y \in \{-1, +1\}$, and performs a linear regression on the observed x and y. In case $n < d$, there is no unique solution that minimizes the squared loss on the training data, in that case the pseudo-inverse is used to find the minimum-norm solution. In case $n \geq d$, the hyperplane with minimal squared loss on the training set is unique. For more detail regarding the terminology and origins of the name of this classifier please see Duin [41] and Bishop [42, p. 186].

4.2.7. FEATURE CURVES AND COMPLEXITY

The word feature refers to the d measurements that constitutes an input vector $x \in \mathbb{R}^d$. A feature curve is obtained by plotting the performance of a machine learning algorithm A against the varying number of features it is trained on [43, 44], see Figure 4.2b for an example. To be a bit more specific, let $\varsigma_{d'}$ be a procedure that selects $d' \leq d$ of the original d features, hence reducing the dimensionality of the data to $d' \in \{1, \ldots, d\}$. More generally, we can also consider procedures that project the data to a lower $d'-$dimensional space (such as PCA). A feature curve is then obtained by plotting $\bar{R}(A(\varsigma_{d'}(S_n)))$ versus d'. As opposed to the learning curve (see Figure 4.2c) where d is kept constant, n is now the quantity that is fixed. As such, it gives a view complementary to the latter.

The selection of d' features as carried out by means of $\varsigma_{d'}$ can be performed in various ways. Sometimes features have some sort of inherent ordering. In other cases principal component analysis (PCA) or feature selection can provide such ordering. When no ordering can be assumed, $\varsigma_{d'}$ samples d' random features from the data—possibly even with replacement. In this scenario, it is sensible to construct a large number of different feature curves, based on different random subsets, and report their average as the final curve.

Typically, an increase in the number of input dimensions means that the complexity of the learner also increases, which may lead to overfitting. This leads to the peaking phenomenon of feature curves (also the peak effect, peaking or Hughes phenomenon [45–48]). This effect is related to the curse of dimensionality and illustrates that adding features may actually degrade the performance of a classifier, leading to the classical U-shaped feature curve. Behavior more complex than the simple U-shape has been observed as well [49–51] and has recently been referred to as double descent [52], see Figure 4.2b. This is closely related to peaking of (ill-behaving) learning curves, this is apparent from Figures 4.2a to 4.2c that show their relation. The causes of this phenomena are discussed in more detail in Subsection 4.6.2. Note that peaking of learning curves unfortunately is a different effect than peaking of feature curves but uses the same terminology.

Feature curves can provide insights into such phenomena. Since dimensionality is often coupled to learner complexity, it may also be interesting to plot performance against the complexity of the learner to obtain conceptually similar curves. Some complexity measures are difficult to calculate or bound, in that case one may settle for an approximate complexity measure. For example, changing hyperparameters of the learner, such as the smoothness of a kernel or the amount of filters in a CNN, can be used to vary the complexity to obtain similar curves [43, 53–56]. Such curves are called *complexity curves* [55], *parameter curves* [57] or *generalization curves* [58].

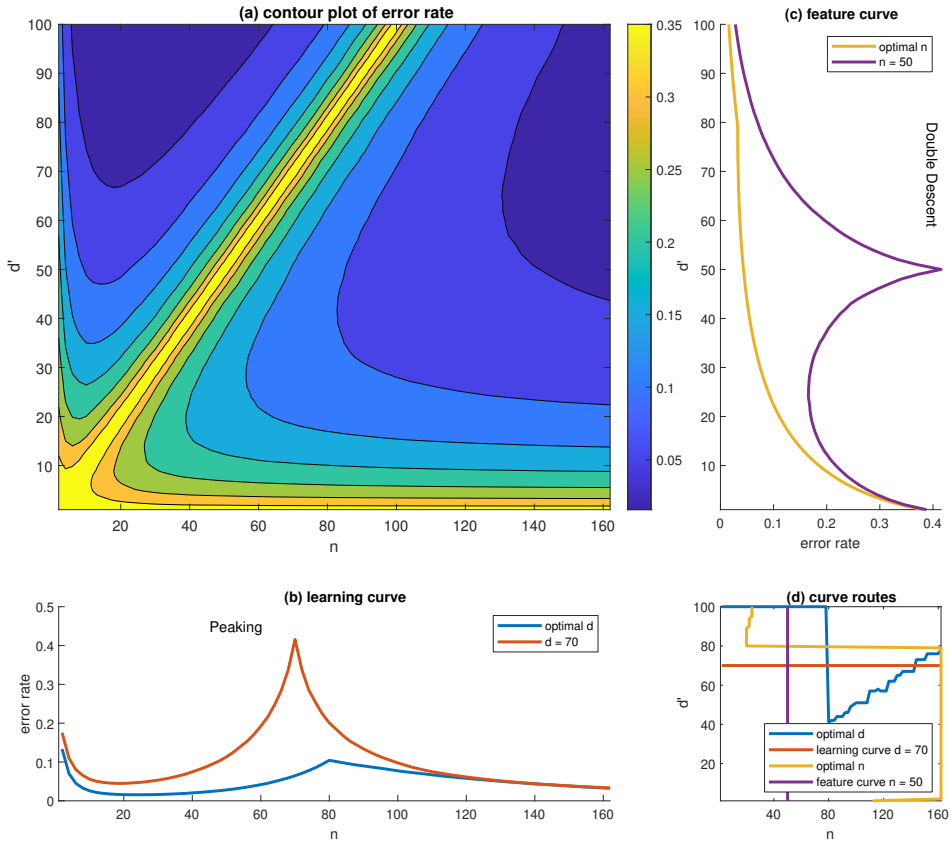


Figure 4.2.: (a) Image of the error for varying sample size n and dimensionality d, for the Pseudo-Fisher learning algorithm (without intercept) on a toy dataset with two Gaussian classes having identity covariance matrices. Their means are a distance of 6 apart in 100 dimensions, with every dimension adding a same amount to the overall distance. We highlight this example to illustrate the relation between peaking and double descent, and to illustrate learning curves versus feature curves. (b) By fixing d and varying n, i.e., taking a horizontal section, we obtain a learning curve (red). We also show the curve where d is chosen optimally for each n (blue). (c) This graph is rotated by 90 degrees. By fixing n and varying d, i.e., a vertical section, we obtain a feature curve (purple). We also show the curve where the optimal sample size n is chosen for each d (yellow). (d) Here we show the paths taken through the image to obtain the learning and feature curves. The learning curve and feature curves are the straight lines, while the curves that optimize n or d take other paths. Observe that the largest n and d (for this considered parameter range) are not always optimal. The cause for these peculiar behavior is related to instability of the learner, see Section 4.6.2 for more detail, but note that related phenomena have been observed for other learners as well, such as deepnets or random forests.

4.2.8. COMBINED FEATURE AND LEARNING CURVES

Generally, the performance of a learner A is not influenced independently by the the number of training samples n and the number of features d. In fact, several theoretical works suggest that the fraction $\alpha = \frac{d}{n}$ is essential [59–61]. Because of the feature-sample interaction, it can be insightful to plot multiple learning curves for a variable number of input dimensions or multiple feature curves for different training set sizes. Another option is to make a 3D plot—e.g., a surface plot—or a 2D image of the performance against both n and d directly. Instead of the number of features we can use any other measure of complexity as well.

As an illustration, Figure 4.2a shows a plot of the performance of Pseudo-Fisher when varying both n and d. Taking a section of this surface, we obtain either a learning curve in Figure 4.2b (horizontal section, fixed d) or a feature curve in Figure 4.2c (vertical section, fixed n). The full, 2D contour plot gives a more complete view of the interaction between n and d. In Figure 4.2d the paths of the learning curves and features curves are shown in the surface, note that we also plot the optimal learning curve (where d is optimized for each n using the surface plot's data) and the optimal feature curve (where n is optimized for each d). Here, we can observe that the optimal d depends on n. Likewise, for the particular classifier that we study, there is an optimal n for each d, i.e., the largest possible value of n is not necessarily the best, especially in case of an ill-behaved learning curve.

Duin [50] is possibly the first to include such 3D plot, though already since the work of Hughes [43], figures that combine multiple learning or feature curves have been used [62, 63]. Learning curves have also been used in combination complexity curves [53, 54, 64, 65]. More recently, Nakkiran *et al.* [56] and Rosenfeld *et al.* [66] gives 2D images of the performance of deep neural networks as a function of both model and training set size.

4.3. GENERAL PRACTICAL USAGE

The study of learning curves has both practical and theoretical value. While we do not necessarily aim to make a very strict separation between the two, more emphasis is put on the latter further on. This section focuses on part of the former and covers the current, most important uses of learning curves when it comes to applications, i.e., model selection and extrapolation to reduce data collection and computational costs. While these applications are promising, especially in the context of big data and deep learning, their applicability is limited due to uncertainty around which parametric models should be used (exponential, power law, etc.) to model learning curves, as will become apparent from the later sections. Therefore, the works covered in this section are rather preliminary, as most works benchmark on quite a limited amount of datasets where curves seem more well-behaved. Only a few studies model learning curves non-parametrically, which offers much more promise as this is more resistant to ill-behaving curves.

4.3.1. BETTER MODEL SELECTION AND CROSSING CURVES

Machine learning as a field has shifted more and more to benchmarking learning algorithms, e.g., in the last 20 years, more than 2000 benchmark datasets have been created (see [67] for an overview). These benchmarks are often set up as competitions [68] and investigate which

algorithms are better or which novel procedure outperforms existing ones [23]. Typically, a single number, summarizing performance, is used as evaluation measure. For example, perhaps the most well-known competition was the MNIST competition [69]. MNIST is a dataset of handwritten digits, and the goal for learning algorithms is to identify the digit given its image. For the competition the accuracy on the testset was used as evaluation measure.

A recent meta-analysis of various benchmarks on the website "Papers With Code" indicates that the most popular measures are accuracy, the F-measure, and precision [70]. An essential issue these metrics ignore is that sample size can have a large influence on the relative ranking of different learning algorithms. For example, one learning algorithm may be ranked quite badly for one particular sample size, but may outperform others if the training dataset is smaller. In a plot of learning curves this would be visible as a crossing of the different curves (see Figure 4.1). In that light, it is beneficial if benchmarks consider multiple sample sizes, to get a better picture of the strengths and weaknesses of the approaches. The learning curve provides a concise picture of this sample size dependent behavior.

Crossing curves have also been referred to as the scissor effect and have been investigated since the 1970s [63, 71–73]). Contrary to such evidence, there are papers that suggest that learning curves do not cross [74, 75]. The first claim by Kohavi [74] is made for C4.5 versus Naive-Bayes on 9 datasets based on empirical observations. The claim is questionable, as Perlich *et al.* [23] shows that sometimes order of magnitude samples are needed to detect the crossing point. The latter claim of Bornschein *et al.* [75] is specific to deep learning, where, perhaps, exceptions may occur that are currently not understood. They find empirically that the ranking of deep learning models stays the same over multiple orders of magnitude for four popular image classification datasets.

Perhaps the most convincing evidence for crossing curves is given by Perlich *et al.* [23]. They compare logistic regression and decision trees on 36 datasets. In 15 of the 36 cases the learning curves cross. This may not always be apparent, however, as large sample sizes may be needed to find the crossing point. In the paper, the complex model (decision tree) is better for large sample sizes, while the simple model (logistic regression) is better for small ones. Similarly, Strang *et al.* [76] performed a large-scale meta-learning study on 294 datasets, comparing linear versus nonlinear models, and found evidence that non-linear methods are better when datasets are large. Ng and Jordan [15] found, when comparing naive Bayes to logistic regression, that in 7 out of 15 datasets considered the learning curves crossed. Besides, there is much more evidence in favor of crossing curves [31, 77–80].

Also using learning curves, Perlich *et al.* [23] finds that, besides sample size, separability of the problem can be an indicator of which algorithm will dominate the other in terms of the learning curve. Beyond that, the learning curve, when plotted together with the training error of the algorithm can be used to detect whether a learner is overfitting [48, 57, 81, 82]. Besides sample size, dimensionality seems also an important factor to determine whether linear or non-linear methods will dominate [76]. To that end, learning curves combined with feature curves may offer further insights.

4.3.2. EXTRAPOLATION TO REDUCE DATA COLLECTION COSTS

When collecting data is time-consuming, difficult, or otherwise expensive the possibility to accurately extrapolate a learner's learning curve can be useful. Extrapolations give an

impression beforehand of how many examples to collect to come to a specific performance and allows one to judge when data collection can be stopped [32]. Examples of such practice can, for instance, be found in machine translation [37] and medical applications [12, 13, 19].

Last [34] quantifies potential savings assuming a fixed cost per collected sample and per generalization error. Extrapolating the learning curve using some labeled data, the point at which it is not worth anymore to label more data can be estimated and data collection stopped. Note that in practice these costs (per sample, per error) are difficult to characterize, and may not even admit a cost per sample or per error. This remains still unexplored in the study of Last, therefore, their approach may seem overconfident in predicting the optimal point to stop labeling.

Determining a minimal sample size that is required to detect a significant difference in statistics is called (minimal) sample size determination. For usual statistical procedures this is done through what is called a power calculation [83]. For classifiers, sample size determination to determine sufficient performance using a power calculation is unfeasible according to Mukherjee *et al.* [12] and Figueroa *et al.* [13]. John and Langley [84] illustrate that a power calculations that ignores the machine learning model indeed fails to accurately predict the minimal sample size necessary to achieve a particular performance. However, the terminology minimal sample size determination has found continued use in the learning curve literature [12, 13]. It basically means to determine the minimum amount of training samples needed to reach a particular performance.

4.3.3. SPEEDING UP TRAINING AND TUNING

Learning curves can be used to reduce computation time and memory with regards to training models, model selection and hyperparameter tuning.

To speed up training, *progressive sampling* Provost *et al.* [24] uses a learning curve to determine if less training data can reach adequate performance. Crucially, is is assumed the learning curve is well behaved, in this context this means that the slope of the curve is monotonically non-increasing for the points sampled along the learning curve, possibly contaminated with some noise. Since accuracy is used in this work instead of error rate, they assume curves are increasing instead of decreasing. They propose that, if the slope of the curve becomes too flat, learning should be stopped, making training potentially much faster. The slope at a point on the curve is estimated by sampling points on the learning curve in a local neighbourhood and performing a linear regression. They recommended to use a geometric schedule for sampling n to reduce computational complexity. Additionally, they sample the learning curve locally and multiple times around the geometric schedule to estimate the slope. This sampling of the curve to estimate the slope instead of a parametric fit may avoid some problems with ill-behaved curves.

Several variations on progressive sampling exist. John and Langley [84] proposes the notion of *probably close enough* where a power-law fit is used to determine if the learner is "epsilon-close" to the asymptotic performance. This in contrast to Provost *et al.* [24], who avoids fitting curve models instead, potentially making their approach more robust to ill-behaved curves. While the previous works took a rather heuristic approach, Meek *et al.* [85] gives a more rigorous decision theoretic treatment of the topic. By assigning costs to computation times and performances, they estimate what should be done to minimize the

expected costs. Nonetheless, various approximations are necessary to come to an efficient algorithm. Progressive sampling also has been adapted to the setting of active learning [86]. Leite and Brazdil [87] combines meta-learning with progressive sampling for a further speedup.

The most promising studies combine progressive sampling with non-parametric meta-learning on learning curves. Meta-learning uses experience on previous datasets to inform decisions on new datasets. Leite and Brazdil [87] builds a small learning curve on a new and unseen dataset and compares it to a database of previously collected learning curves to determine the minimum sample size. Crucially, they do not assume a particular parametric form for the curve, but instead find the nearest-neighbour learning curve, and are one of the few studies to run a benchmark on a larger scale (60 datasets), as such this is one of the most promising works. They find that they can predict the stopping point by just sampling 4 points on the learning curve, leading to a significant speed up of factor 12 with an acceptable error. In a follow-up work, Leite and Brazdil [88] extends the approach to predict which of the two classifiers will perform best on a new dataset. This can be used to avoid costly evaluations using cross validation. Leite and Brazdil [89] go even further and propose an iterative process that predicts the required sample sizes, builds learning curves, and updates the performance estimates in order to compare two classifiers. van Rijn *et al.* [90] extend the technique to rank many machine learning models according to their predicted performance, and tune their approach to come to an acceptable answer in as little time as possible.

With regards to hyperparameter tuning, already in 1994 Cortes *et al.* [31] devised an extrapolation scheme for learning curves, based on the fitting of power laws, to determine if it is worth to fully train a neural network. In the deep learning era, this has received renewed attention. Hestness *et al.* [91] extrapolates the learning curve as a power law to optimize hyperparameters. Hoiem *et al.* [30] takes this a step further and actually optimize several design choices, such as data augmentation, but also assume a power-law, which however seems appropriate for deep learning in some cases, as we will discuss in the next section.

4.4. EMPIRICAL WORKS THAT FAVOR WELL-BEHAVED CURVES

We deem a learning curve well-behaved if it shows improved performance with increased training sample sizes, i.e., $\bar{R}_n(A) \geq \bar{R}_{n+1}(A)$ for all n. Learners that satisfy this property are called smart [2, page 106] and monotone [3]. There is both experimental and theoretical evidence for well-behaved curves.

In this section we discuss empirical studies of the shape of learning curves. The shape is usually assumed to have a particular parametric form, such as exponential or power law. We discuss the parametric forms used to model learning curves, and we make some surprising observations, for example that some curve models are redundant. Then we discuss empirical studies that fit learning curves with these parametric models in Section 4.4.2. We mention which studies have which previously mentioned curve fitting pitfalls (see Section 4.2.5). Results for shallow learners are mostly inconclusive or contradicting, probably due to these pitfalls. Two recent and more trustworthy studies, Mohr *et al.* [40] and Brumen *et al.* [39], do not contradict each other and do give some clearer picture of the overall best

parametric model. Evidence for deep nets is most convincing in favor of the power law, but requires sufficient tuning of hyperparameters such as learning rate and network size, otherwise deviations occur. In the section after the current, we turn to learning theoretical works in favor of well-behaved curves and their implications for the shape.

4.4.1. ANALYSIS OF PARAMETRIC LEARNING CURVE MODELS

Reference	$y(n)$	Example constraints	Limit	+ ?	Used in
POW2	an^{-k}	$a>0, k>0$	0	☑	[32][36][33][34]
POW3	$an^{-k}+c$	$a>0, c>0, k>0$	c	☑	[36][37][31][92][39]
POW4	$a(n+\delta n)^{-k}+c$	POW4 = POW3, so see above			[37]
LOG2	$-a\log(n)+c$	$a>0$	$-\infty$	□	[33][32][36][34][92][39]
EXP2	ab^n	$a>0, 0<b<1$	0	☑	[32][33][34]
EXP3	ab^n+c	$a>0, 0<b<1, c>0$	c	☑	[92][37][39]
EXPD3	$c-(c-a)b^n$	EXPD3 = EXP3, so see above			[37]
EXP4	$ab^{(n^k)}+c$	$a>0, 0<b<1, c>0, k>0$ $a>0, b>1, c>0, k<0$	c c	☑ ☑	[37]
EXPP3	$b^{((n-\delta n)^k)}+c$	same constraints as above			[37]
WBL4	$c-be^{-an^k}$	WBL4 = EXP4, so see above			[36]
LIN2	$-an+c$	$a>0$	$-\infty$	□	[32][33][34][92]
ILOG2	$a/\log(n)+c$	$a>0, c>0$	c	☑	[37]
VAP3	$ab^{1/n}n^{-k}$	$a>0, b>0, k>0$	0	☑	[36]
MMF4	$(ab+cn^k)/(b+n^k)$	$a>0, b>0, c>0, k>0$ $a>0, b>0, c>0, k<0$	c a	☑ ☑	[36]

Table 4.1.: Parametric forms that model the risk, $\bar{R}_n(A) \approx y(n)$. n indicates the training set size, parameters are: a a multiplicative constant, b the base(s), c offset / bias term, δn translation of the x-axis, k the exponent. Various constraints are plausible and here we give some plausible options (!). If possible, we give constraints that lead to finite and positive curves suitable for extrapolation to larger n. If a parameter is not mentioned under constraints it is in $\mathbb{R}$. + (Positive) indicates whether the $y(n) > 0$ for all $n > 0$ and for all parameter values that satisfy the constraints. Limit is $\lim_{n\to\infty} y(n)$ given the constraints. Note that one can use the identity $b^n = e^{\log(b)n}$ to rewrite the EXP* formulas in terms of the exponential function. Mohr *et al.* [40] uses all models above.

Various works have studied the fitting of empirical learning curves and found that they typically can be modelled with function classes depending on few parameters. Table 4.1 provides an overview of the common parametric models for fitting learning curves in machine learning and the studies which use them. Models for training curves [93] and human learning [94] may offer further candidates.

Let us first discuss several issues related to the parametric learning curve models. Prior

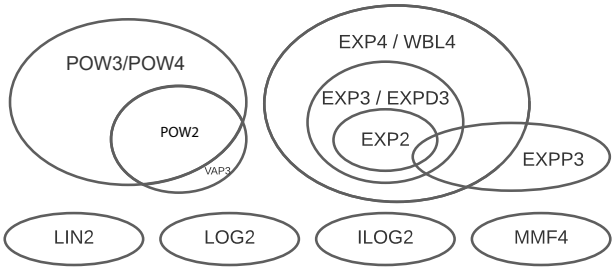


Figure 4.3.: Overview of curve model relations. Some curve models are equivalent (indicated by the slash), and others have some overlap for particular parameter choices.

works usually do not give any suggestions for the ranges of parameter values for the curve models (a, b, c, etc.), except Brumen *et al.* [92] and Gu *et al.* [36] who give the range for some parametric forms. We focus on extrapolation, and we give some possibly sensible parameter ranges for all the curve models for that purpose. That is, we give parameter ranges that lead to finite positive risk in the limit of infinite training samples and non-negative risks. Note that we do not give all exhaustive options that satisfy the above. In some cases it is not possible to satisfy these contraints, as indicated in the column 'Limit' and '+?' in Table 4.1. In light of Section 4.6, strictly decreasing risk may be too much to ask for, which is why we focus on non-negativity and limiting behavior.

We want to emphasize that these are one of the many possible choices for the parameter ranges. Especially if one wants to interpolate a learning curve, one could make different choices, as limiting behavior or non-negativity everywhere might not be important. For example, if $a < 0$ and $k < 0$ for POW3, the curve can be non-negative up to a certain value of n. This allows for more flexibility in fitting curves, because for $a < 0$ and $k < 0$ the curve is concave instead of convex. But since this results in the limit in a negative and unbounded risk it is not in the table.

The parameter space may be further pruned if bounded losses (such as accuracy) are used. However, one should be careful: pruning the parameter space may come at a non-negligible decrease in flexibility of curve models. This is especially apparent for LIN2; this model would be completely ruled out for fitting if we demand its range is always in $[0,1]$ for all n. In that case the only possible fit satisfying the constraints is $y = a$ where $b = 0$. Meanwhile, LIN2 could model parts of the learning curve well, especially if it seems the learner is not making progress (see Figure 4.4a). Therefore, the parameter ranges for all parametric models should be determined with care, otherwise fitting performance may suffer unnecessarily.

Even for extrapolation, it is not completely clear cut that non-negative and non-zero risk in the limit is required. Especially when extrapolating from n to $n+1$, this might be irrelevant, as the limiting behavior may not be noticeable in the extrapolation. However, for extrapolating to a sample size n that is much larger than the current, non-negative and non-

zero risk in the limit will likely be more important. For example, LIN2 and LOG2 are then clearly undesirable, because for large enough sample sizes they will make negative predictions. Similarly, parametric models that have a limiting value of 0 seem unsuitable, since for many difficult real-world problems often the Bayes-error will be non-zero. However, currently it remains unclear when non-negative risks and non-zero risk in the limit is a strong requirement, for this a deeper empirical investigation is necessary. This pitfall regarding the bias term was also discussed earlier in Section 4.2.5.

One should be aware that some curve models have discontinuities. For example, ILOG2 has a discontinuity at $n = 1$, and MMF4 has a discontinuity at $n = -(b)^{1/\alpha}$. Additionally, one may consider to rescale n to the range $(0, 1]$ by dividing n by the maximum size of the training set before fitting. This affects the shape of the curve model, for example, for $0 < n < 1$ ILOG2 is concave and for $n > 1$ ILOG2 is convex. Current studies that fit curves, we believe, do take $n \in \mathbb{N}$ but usually do not mention this. Besides normalizing n to $(0, 1]$, other transformations of the x-axis or y-axis could also be considered; such as log-transformations. Such transformations can increase the class of functions for fitting even further.

Another unexplored aspect of the parametric curve models is their overlap and modeling power, some preliminary results are visualized in Figure 4.3. Rewriting analytically some models, we find that POW3 and POW4, EXP4 and WBL4, EXP3 and EXPD3, are all in fact the same model family. Or in other words, the parameters can be set in such a way that one parametric model can be analytically rewritten to another. Note that for this analysis, we have ignored the given constraints. Curve fitting studies did not notice this redundancy in parametric models in comparative studies; instead Mohr *et al.* [40] find significant differences in their performance. This can be explained by the fact that due to a different parametrization, parameters are initialised in a different way, and curve fitting procedures act different, yielding alternative results. Note that by setting parameters to specific values, we can also find overlap between classes. For example, by setting $b = 1$ for VAP3, we find VAP3 has overlap with POW2, and this overlap is contained further in the class POW3/POW4. Furthermore, for EXPP3, we can set $\delta n = 0$. If we then set $a = 1$ for EXP2, EXP3 or EXP4/WBL4, we find models that are also in the restricted model class of EXPP3.

To deepen our understanding further, one could generate learning curves with one mathematical model and try to fit it with an other. Since anyway a fit is never perfect (thus analytical rewriting is not the whole story), this can give deeper insight which classes have more modeling power than others. One could also further study this analytically, by, for example, approximating curve models for large n.

Study	Learning curve models compared	Learners	#D	E?	B?	F?	S?
Frey and Fisher [32]	POW2, LIN, LOG2, EXP2	C4.5	14	☑	□	?	☑
Gu *et al.* [36]	POW2, POW3, LOG2, VAP, MMF, WBL	LR, C4.5	8	☑	☑	☑	□
Kolachina *et al.* [37]	EXP3, EXP4, EXPP3, POW3, POW4, ILOG2	Moses SMT	30	☑	☑	☑	□
Last [34]	POW2, LIN, LOG2, EXP	IN	6	□	□	?	□
Boonyanunta and Zeephongsekul [95]	EXPDIFF	LDA, LR, NN, KNN, C4.5	3	☑	-	?	-
Singh [33]	POW2, LIN, LOG2, EXP	KNN, RBF SVM, NN, C4.5	4	□	□	□	□
Ahmad and Tesauro [96]	EXP2	perceptron	1 (A)	□	□	□	□
Cohn and Tesauro [35]	EXP2, POW2	perceptron	4 (A)	□	□	□	□
Brumen *et al.* [92]	POW3, LIN, LOG2, EXP3	C4.5	121	□	☑	☑	☑
Brumen *et al.* [39]	POW3, EXP3, LOG2	C4.5, NN, NB, SVM	130	☑	☑	☑	☑
Mohr *et al.* [40]	all of Table 4.1 (MMF4,WBL4)	20 learners	246	☑	☑	☑	☑

Table 4.2.: Studies comparing parametric models, overall best are in underlined. Abbreviations: #D (Number of datasets, A means artificial data), E? (Extrapolation?), B? (Bias terms?), F? (Nonlinear Fitting?), S? (Statistics?). Options: ☑/□/?/- (Yes/No/Unclear/Irrelevant, Yes indicates better quality study). Abbreviations for learners: C4.5 (a decision tree algorithm), LR (logistic regression), IN (information network - a decision tree algorithm), Moses SMT (machine translation system), LDA (Fisher's linear discriminant), NN (neural network), KNN (K-nearest neighbours), RBF SVM (radial basis kernel support vector machine), NB (Naive Bayes).

4.4.2. EMPIRICAL LEARNING CURVE FITTING STUDIES

Table 4.2 provides an overview of all the curve studies. Many studies reach contradicting conclusions. Our hypothesis is that this is due to the fact that some studies make common mistakes in their experimental design. We have mentioned these common pitfalls in Section 4.2.5 and have indicated them withe the letters E (Extrapolation?), B (Bias term?), F (Nonlinear Fitting?) and S (Statistics?). In the table, an overview is given of the various studies, the curve models they compare, the learners, amount of datasets, and their potential pitfalls. An additional complicating factor is that the experimental setups and fitting procedures are not uniform, or sometimes not sufficiently described, leading to a '?' in the table. Observe that the studies compare various different learners and parametric models, making an overall comparison cumbersome. Some studies look only at few datasets, which also calls into question the generality of their results.

Let us now discuss the studies in more detail by traversing the table from top to bottom. Frey and Fisher [32] find POW2 to be the best for decision trees, but do not include bias terms, calling their results into question. Gu *et al.* [36] extend this work to more datasets and learners. They use an offset in their power law and consider other functional forms, notably, VAP3, MMF4, and WBL4. For extrapolation, no model consistently makes the best predictions, but based on the rankings of the curve models with respect to the mean squared error, POW3 performs best on average over all experiments. They however do not evaluate the significance of this finding. Kolachina *et al.* [37] perform a fairly large study with decent quality for machine translation. They find that POW3 achieves the lowest average root mean squared error for predicting the next point or the last point on the curve, but is not always (systematically) better. However, they also do not quantify the significance.

Last [34] focusses more on applications, but do also evaluate curve fitting. However, the number of datasets is small and their study suffers from all pitfalls, but find POW2 to be the best. Boonyanunta and Zeephongsekul [95] come up with the novel idea that a differential

equation should model learning curves (an idea whose potential remains unexplored), leading them to an exponential form, indicated by EXPD3 in the table. However, it turns out that this curve model can be rewritten to EXP3, and they further do not perform a quantitative analysis comparing their model to others. Singh [33] evaluate 4 learners on 4 datasets, and find LOG2 provides the best fit. The authors have some reservations about their results, and remark that it may be an artefact of their curve fitting procedure (fitting of log values). They also suffer from all pitfalls, and we remark that n was never larger than 2000 samples, which may offer another reason why LOG2 could work well. For large n the performance must necessarily suffer as the LOG2 model would diverge.

One of the few shape studies with synthetic data are those by Ahmad and Tesauro [96] and Cohn and Tesauro [35]. Since most empirical studies deal with datasets of finite size, learning curves are usually constructed until all data is used up. For artificial problems, it is unclear up until what point learning curves should be created, since there is no upper bound on n. One may consider to stop creating the learning curve if a learner seems converged, but detecting convergence robustly is difficult (see Section 4.3.3) and may be impossible in light of the ill-behaving curves discussed in this survey (see Figure 4.4). Ahmad and Tesauro [96] and Cohn and Tesauro [35] fix a maximum training set size arbitrarily, we do not know of an alternative and consider this an open problem. Considering fitting performance in the limit of infinite data is another problem yet to be studied, but poses all kinds of technical difficulties because the mean squared error may diverge.

Ahmad and Tesauro [96] find exponential behavior of the learning curve for a perceptron trained with backpropagation on a toy problem with binary inputs, but do not perform any comparison with the power law. Cohn and Tesauro [35] extends their work to four synthetic datasets and compare the learning curves using the R^2 goodness of fit (which has its problems, see Section 4.2.5). Two synthetic problems are linearly separable, the others require a hidden layer, and all can be modeled perfectly by the neural network. Whether a problem was linearly separable does not seem to matter for the shape. For binary features the curves seem to be more of an exponential type, whereas real-valued features seem to be slightly better modeled by a power law (which agrees with theory, see Section 4.5.1). However, they also note, that it is not always clear that one fit is significantly better than the other and their setup is not reproduceable. So while this seems to agree with theory, their findings warrant further investigation.

Brumen *et al.* [92] considers only the performance of the fit on already observed learning curves points. They *do* only use learning curve models with a maximum of three parameters, possibly alleviating overfitting concerns a little. They employ a total of 121 datasets and use C4.5 for learning. In 86 cases, the learning curve shape fits well with one of the functional forms, in 64 cases EXP3 gave the lowest overall MSE, in 13 it was POW3. A signed rank test shows that the exponential outperforms LIN2, LOG2, POW3 significantly ($p < 0.001$), providing decent evidence that decision trees more often behave as exponentials.

Brumen *et al.* [39] further extend their own study to more learners and datasets. However, curiously, they sample less values for n to make curves more well-behaved. In our opinion, the more n are sampled along a learning curve, the better, as we have more points for fitting the curve and evaluating the fit, which should make the study more robust. They further improve their experimental design to use stratification which is sensible, but also claim that k-fold cross validation results in more ill-behaved learning curves which we think is

questionable. They instead resort to a single 80/20 split. It should be apparent that more test sets will lead to better learning curve estimates (because a single test set could be a poor sample), but perhaps their procedure used a different test set per n, which would explain their observed ill-behaving curves when using k-fold. We suggest to use a fixed test set for different n to avoid this issue. A positive innovation is that they consider how many points were used for fitting learning curves (25%, 50%, 75%) to analyse their results. They find that the fitting performance is not normally distributed and perform a careful statistical analysis to determine the significance of their results, in addition it is explained what happens to failed fits. If 25% points are used for fitting, the power law and exponential tie, otherwise the power law performs significantly better than the EXP3 and LOG2. Their conclusions curiously contradict their own earlier work [92], which claims that exponentials model learning curves better for the C4.5 algorithm. No reflection is given why this might be the case.

Recently Mohr *et al.* [40] performed the largest scale study on learning curves, evaluating as many as 20 learners on 246 datasets. They publish their curve data in a publicly accessible API ensuring reproducibility, and this will make it easy for the community to further investigate various learning curve questions. They include all curve models of Table 4.1, and use 5 different test sets. They do not build monotonically increasing training sets as they focus on model selection purposes. They average curves over over 5 different training and 5 test sets, leading to 25 learning curves which are averaged. This averaging makes the curves more well-behaved and the fitting easier. However, they still ran into fitting issues. To deal with this they fit parametric models 5 times and use several criteria to discard bad fits. Like Brumen *et al.* [39], Mohr *et al.* [40] find that the mean squared error is not normally distributed, and they use a similar evaluation to determine significance. Surprisingly, they find that MMF4 and WBL4 can achieve the best results (on average over learners and datasets) compared to the other parametric models of the table, when sufficient learning curve points are used for fitting the curve model. Considering the bias-variance trade-off this is perhaps not surprising — more parameters in these curve models can be efficiently utilized if enough curve data is available to set them correctly. In light of the previous subsection, it should be noted that WBL4 is the same as EXP4, and thus again we find an apparent contradiction with [39] who found that the power law performs better than the exponential. However, this contradiction can be explained by the fact Brumen *et al.* [39] only considered a limited set of curve models, while Mohr *et al.* [40] considers many more parametric models. The findings of Mohr *et al.* [40] for the LOG2, EXP3 and POW3 do agree with those of Brumen *et al.* [39]. Mohr *et al.* [40] also considers a trivial baseline which predicts the last seen point on the curve which is surprisingly competitive with the other parametric models in terms of extrapolation performance, further emphasizing the difficulties of learning curve fitting.

Several curve fitting studies mention that curve models usually do not outperform others systematically [33, 36, 37]. This may indicate that a universal parametric learning curve model may be too much to ask for. This is further underscored by the various ill-behaved shapes in Section 4.6. However, we can still investigate what determines the parameters of the fits or what factors influence the shape. Mukherjee *et al.* [12], Cortes *et al.* [31], and Hoiem *et al.* [30] wonder if the asymptotic value of the power law and its exponent could be related; in other words, they think that the learning speed is higher (smaller b for EXP3)

if the asymptotic value is smaller (smaller c). Singh [33] investigates the relation between dataset and classifier but does not find any interaction effect. They *do* find that the neural networks and the SVM are more often well-described by a power law and that decision trees are best predicted by a logarithmic model. Perlich *et al.* [23] speculate that the Bayes error may be indicative of whether the curves of decision trees and logistic regression will cross or not. In case the Bayes error is small, decision trees will often be superior for large sample sizes. All in all, there are few results of this type and most are quite preliminary. The database of Mohr *et al.* [40] enables the community to investigate these issues in more depth.

4.4.3. POWER LAWS AND EYE-BALLING DEEP NET RESULTS

Studies of learning curves of deep neural networks mostly claim to find power-law behavior. However, initial works offer no quantitative comparisons to other parametric forms and only consider plots of the results, calling into question the reliability of such claims. Later contributions find power-law behavior over many orders of magnitude of data, offering substantially stronger empirical evidence.

Sun *et al.* [97] state that their mAP@ performance on a large-scale internal Google image dataset increases logarithmically in dataset size. This claim is called into question by Hestness *et al.* [91] and we would agree: there seems to be little reason to believe this increase follows a logarithm. Like for Singh [33], who also found that the logarithm fit well, one should anyway remark that the performance in terms of mAP@ is always bounded from above and therefore the log model should eventually break. As opposed to Sun *et al.* [97], Joulin *et al.* [98] creates a learning curve up to a training set size of 100 million images, and qualitatively observes an effect of diminishing returns in the empirical learning curve. Mahajan *et al.* [99] also studies large-scale image classification and find learning curves that level off more clearly in terms of accuracy over almost 3 orders of magnitudes (training set sizes from 0.5×10^7 to 3×10^9). They presume that this is due to the maximum accuracy being reached, but note that this cannot explain the observations on all datasets. In the absence of any quantitative analysis (such as fitting of parametric models or plots on log-log scale), these results are possibly not more than suggestive of power-law behavior.

The first to offer strong convincing empirical evidence for the power law are [91]. They show power laws over multiple orders of magnitude of training set sizes for a broad range of domains: machine translation (error rate), language modeling (cross entropy), image recognition (top-1 and top-5 error rate, cross entropy) and speech recognition (error rate). The exponent was found to be between -0.07 and -0.35 and mostly depends on the domain. Architecture and optimizer primarily determine the multiplicative constant. For small sample sizes, the power law supposedly does not hold anymore, as the neural network converges to a random guessing solution. Overall, the power law behavior turns out to be so robust that they suggest one can search for new architectures at smaller sample sizes to speed up experiments. To uncover the power law, significant tuning of the hyperparameters and model size per sample size is necessary, otherwise deviations occur. Hoiem *et al.* [30] investigate robust curve fitting for the error rate using the power law with offset, and use extrapolations to optimize design decisions. They generally find exponents of size -0.3 and -0.7, thus of larger magnitude than Hestness *et al.* [91].

Kaplan *et al.* [100] and Rosenfeld *et al.* [66] further corroborate power laws for image

classification and natural language processing. Kaplan *et al.* [100] finds that if the model size and computation time are increased together with the sample size sufficiently, that the learning curve has this particular behavior. If either is too small this pattern disappears. They find that the test loss also behaves as a power law as function of model size and training time and that the training loss can also be modeled in this way. Rosenfeld *et al.* [66] reports that the test loss behaves as a power law in sample size when model size is fixed and vice versa. Both provide models of the generalization error that can be used to extrapolate performances to unseen sample and model sizes and that may reduce the amount of tuning required to get to optimal learning curves. They also propose a model for the transition of a power law to random guessing performance.

4.5. LEARNING THEORY IN FAVOR OF WELL-BEHAVED CURVES

In this section we discuss implications of learning theory studies on the shape of learning curves. Theory gives some indication that properties of the problem (such as separability or realizeability) and the complexity of the model class determine whether the curve will be of an exponential or a power-law shape. However, most theory does not apply to learners or settings considered in practice. In particular, we also cover Probably Approximately Correct (PAC) learning and why it does not tell us much about the shape of the curve. Finally, in literature, various assumptions have been made on the problem, such as loss, separability, etc. that lead to exponential and power-law shapes. PA curves also turn out to be provably monotone if the problem is well-specified, meaning the assumptions on the data generation process are correct, and a Bayesian approach is used. One should note that the various assumptions in this section may be quite strong, yet they give insight into various curve behaviors.

4.5.1. SHAPE DEPENDS ON HYPOTHESIS CLASS

In the context of classification, results from learning theory, especially in the form of Probably Approximately Correct (PAC) bounds [101, 102], have been referred to to justify power-law shapes in both the separable and non-separable case [30]. See Section 1.2 (page 3) for a brief recap of PAC learning, for more detail refer to [102].

The PAC model is, however, pessimistic since it considers the performance on a worst-case distribution P, while the behavior on the actual fixed P can be much more favorable (see, for instance, [103–107]). Even more problematic, the worst-case distribution considered by PAC is *not* fixed, and can depend on the training size n, while for the learning curve P is fixed. As a consequence, it is possible that a PAC bound can say that the risk decreases as $\frac{1}{n}$, while the decrease of the learning curve is exponential [108].

Recently, Bousquet *et al.* [108] gave a full characterization of all learning curve bound shapes for the realizable case and optimal learners. The risk bounds have a form $\bar{R}_n(A) \leq CR'(cn)$, where c and C are constants that can depend on P. The risk bounds more closely capture the learning curve shape, as the behavior of the risk is considered for a single fixed distribution P that cannot change when varying the sample size. They show that risk bounds for optimal learners can have only three shapes: an exponential shape ($R'(n) = e^{-n}$), a

power-law shape ($R'(n) = \frac{1}{n}$), or there is no possible risk bound of this form, a situation that they call arbitrary slow learning. The shape of the bound is determined by novel properties of the hypothesis class (not the VC dimension). The case of no risk bound or arbitrary slow learning means that, when using particular hypothesis classes, for any function $f(n)$ that converges to zero arbitrarily slowly and any learner A, we can find a distribution for which that learner satisfies $\bar{R}_n(A) \geq f(n)$. Basically, it means that for such hypothesis classes, we cannot give an upperbound of the form above in terms of R'. The result of arbitrarily slow learning is, partially, a refinement of the no free lunch theorem of [2, Section 7.2] and concerns, for example, hypothesis classes that encompass all measurable functions. These results are a strengthening of a much earlier result by Cover [109]. Interestingly, the considered optimal learners by Bousquet *et al.* [108] are not empirical risk minimizers, but are a carefully constructed ensemble of online learners. In practical application and empirical studies, different learners are used. Because of this reason, the results do not directly transfer to practical settings. What the consequences are for typical learning curves are in practice yet remains to be explored, but these works appear to be the one of the most relevant regarding the shape of learning curves.

4.5.2. SHAPE DEPENDS ON THE PROBLEM (PA AND NON-PA)

There are a number of works that find evidence for the power-law shape or exponential shape for PA and non-PA learning curves under various assumptions relating to the learning problem. A first provably exponential learning curve can be traced back to the famous work on 1NN [110] by Cover and Hart. They point out that, in a two-class problem, if the classes are far enough apart, 1NN only misclassifies samples from one class if all n training samples are from the other (crucially, sampling should not be stratified for this example). In case both classes have equal prior, one can determine that $\bar{R}_n(A_{1\text{NN}}) = 2^{-n}$. This seems to suggest that if a problem is well-separated, classifiers may converge exponentially fast.

For a classification problem where the classes overlap, Peterson [111] showed for the nearest neighbor classifier (1NN) that in a two-class problem where P_X is uniform on $[0, 1]$ and $P_{Y|X}(1|x) = x$ and $P_{Y|X}(-1|x) = 1 - x$, the learning curve equals

$$\bar{R}_n(A_{1\text{NN}}) = \frac{1}{3} + \frac{3n+5}{2(n+1)(n+2)(n+3)}, \tag{4.4}$$

Amari [112, 113] studies the learning curves for a basic algorithm, the Gibbs learning algorithm, in terms of the entropic error. The entropic error is defined as the negative log of the accuracy. He assumes a binary classification setting, thus $y \in \{-1, +1\}$. The Gibbs algorithm for classification works as follows. Assume a linear classifier for simplicity, thus $\hat{y} = \text{sgn}(w^T x)$ (note that Amari considers more general machines and neural networks). Put a prior probability density on w, thus $w \sim P_W$, which is determined before any data is observed by the learner. Now when a training set S_n is received, we sample $w \sim P_W$ and check its error on S_n. Only if it makes zero error, w is returned, otherwise we keep sampling $w \sim P_W$ until a classifier with zero error is found [61, 114]. It is assumed the problem is separable (otherwise there is no solution and the learner cannot learn) and the model samples are therefore taken from version space [61]. Since the results of Amari are independent of the groundtruth model that generates the labels, his results can be interpreted both as PA or non-PA learning curves.

For the Gibbs learner A_G, the risk in terms of the entropic error can be shown to decompose using a property of the conditional probability [112]. Let $p(S_n)$ be the probability of selecting a classifier from the prior that classifies all samples in S_n correctly. Assume, in addition, that $S_n \subset S_{n+1}$. Assuming S_n, S_{n+1} are given, and that Gibbs is trained on a sample size of S_n, we denote the probability of a correct classification by Gibbs on an unseen sample as $p(y = \hat{y}|S_n, S_{n+1})$ conditioned on the data. Note that randomness in $p(y = \hat{y}|S_n, S_{n+1})$ stems from the fact of the randomness in the Gibbs algorithm. Due to the definition of conditional probability (and due to Gibbs not seeing the $(n+1)$th sample), we have that

$$p(y = \hat{y}|S_n, S_{n+1}) = \frac{p(S_{n+1})}{p(S_n)}. \tag{4.5}$$

Of this expression, still an expectation should be carried out with respect to S_n and S_{n+1} to come to the risk. However, we now have accuracy instead of error rate, because this is formulated in gains instead of losses. Note that taking the expectation w.r.t. S_{n+1} is enough, since S_n is contained in S_{n+1}. The resulting expression will be hard to analyze for general losses [115]. To get around this, Amari takes the loss to be the entropic error, defined as

$$e(S_n, S_{n+1}) = -\log(p(y = \hat{y}|S_n, S_{n+1})), \tag{4.6}$$

which we can indeed interpret as the negative log of the accuracy. Thus this is again a loss function — the lower, the better. Note that this is not equal to the cross entropy loss. For the entropic error the risk is given by

$$\mathbb{E}_{S_{n+1}} e(S_n, S_{n+1}) = \mathbb{E}_{S_n} \log(p(S_n)) - \mathbb{E}_{S_{n+1}} \log(p(S_{n+1})) \tag{4.7}$$

and thus the expression has simplified into the difference of two expectations [112]. These individual expectations are much more tractable to analyze analytically than the expectation of a fraction.

Under some additional assumptions, which ensure that the prior is not singular (meaning all parameters of the model influence its predictions), the behavior asymptotic in n can be fully characterized using an approach similar to the replica method often used in statistical physics. Using this machinery Amari [112] then proves that

$$\mathbb{E}_{S_{n+1}} e(S_n, S_{n+1}) \approx \frac{d}{n} + o\left(\frac{1}{n}\right), \tag{4.8}$$

where d is the number of parameters. Amari claims that the assumption above holds for many machines, including multilayer neural networks. However, we believe this may not be the case for modern neural nets that oftne use the ReLU activation function. Since for a ReLU, if the activation is zero, changes in the parameters will not cause a corresponding change in the network output. However, ReLU's were introduced only around 2011, much later than Amari's works which were published in the nineties, and Gibbs is not used for training deep networks anyhow. But for the sigmoid or tanh activation functions which were used at the time, the assumption is reasonable.

Amari and Murata [116] extend this work and consider labels generated by a noisy process, allowing for class overlap. Instead of a deterministic learner, they now consider a more typical logistic regression model common in machine learning which gives class probability estimates. This work, while closely related to the previous, seems to only consider

the regular learning curve instead of the PA-curve, since the results now do depend on the groundtruth model that generates the labels. Besides Gibbs, they study algorithms based on maximum likelihood estimation and the Bayes posterior distribution. To derive their results, they use that the maximum likelihood and other estimators are asymptotically normally distributed (with variance given by the Fisher Information), and seem reminiscent of a typical asymptotic statistical analysis. They find for Bayes and maximum likelihood that the entropic generalization error behaves as $H_0 + \frac{d}{2n}$, while the training error behaves as $H_0 - \frac{d}{2n}$, where H_0 is the best possible cross entropy loss. For Gibbs, the error behaves as $H_0 + \frac{d}{n}$, and the training error as H_0. In case of model mismatch, the maximum likelihood solution can also be analyzed. In that setting, the number of parameters d changes to a quantity indicating the number of effective parameters, which depends on the curvature of the loss around the model closest to the groundtruth in terms of KL-divergence, and H_0 becomes the loss of that model.

In a similar vein, Amari *et al.* [115] analyze the 01 loss, i.e., the error rate, under the annealed approximation [60, 61], which uses

$$\mathbb{E}_{S_{n+1}} p(y = \hat{y}|S_n, S_{n+1}) \approx \frac{\mathbb{E}_{S_{n+1}} p(S_{n+1})}{\mathbb{E}_{S_n} p(S_n)}. \tag{4.9}$$

This approximation takes the expectation of the numerator and denominator separately, an approximation that Amari earlier tried to avoid. One thus may wonder about the accuracy of this approximation. Four settings are considered, two of which are similar to the previous works [112, 113, 116]. The variation in them stems from differences in assumptions about how the labeling is realized, ranging from a unique, completely deterministic labeling function to multiple, stochastic labelings. Possibly the most interesting result is for the realizable case where multiple parameter settings give the correct outcome or, more precisely, where this set has nonzero measure. In that case, the asymptotic behavior is described as a power law with an exponent of -2. Note that this is essentially faster than what the typical PAC bounds can provide, which are exponents of -1 and $-\frac{1}{2}$. This possibility of a more rich analysis is sometimes mentioned as one of the reasons for studying learning curves [61, 106, 117].

For some settings, exact results can be obtained because symmetries can be exploited. If one considers a 2D input space where the marginal P_X is a Gaussian distribution with mean zero and identity covariance, and one assumes a uniform prior over the true linear labeling function without noise, the PA curve for the zero one loss can exactly be computed to be of the form $\frac{2}{3n}$, while, the annealed approximation gives $\frac{1}{n}$ [115]. Thus the annealed approximation for this example at least gives the correct rate of decrease of the learning curve.

4.5.3. MONOTONE SHAPE IF WELL-SPECIFIED (PA)

The PA learning curve is monotone if the model is well-specified and Bayesian inference is employed. Well specified means that the prior assumed by the model, is the same as the prior that underlies the problem average. The likelihood function, which is used by the model for learning, should also correspond to the likelihood function which generates the data. Well-specified does not make learning trivial: we only know there is a model from our model family which generates the data, but we do not know which model this is.

That well-specification leads to monotone PA curves with a Bayesian approach, is a consequence of the total evidence theorem [118–120]. It states, informally, that one obtains the maximum expected utility by taking into account all observations. However, a monotone PA curve does not rule out that the learning curve for individual problems can go up, even if the problem is well-specified, as the work covered in Section 4.6.6 will show. Thus, if we only evaluate in terms of the learning curve of a single problem, using all data is not always the rational strategy. Therefore, in such a case, we may want to use a technique that tries to evaluate whether more data will lead to improved models, which we will discuss in Section 4.6.7.

Of course, in reality, our model probably has some misspecification, which is a situation that has been considered for Bayesian linear regression and Gaussian Processes, the latter we do not cover in this chapter (see [1]). See Subsection 4.6.5 for unexpected behavior for Bayesian linear regression.

4.6. ILL-BEHAVED LEARNING CURVES

It is important to understand that learning curves do not always behave well and that this is not necessarily an artifact of the finite sample or the way an experiment is set up. Deterioration with more training data can obviously occur when considering the curve $R(A(S_n))$ for a particular training set, because for every n, we can be unlucky with our draw of S_n. That ill-behavior can also occur in expectation, i.e., for $\bar{R}_n(A)$, may be less obvious.

In the authors' experience, most researchers expect improved performance of their learner with more data. Less anecdotal evidence can be found in literature. Shalev-Shwartz and Ben-David [102, page 153] states that when n surpasses the VC-dimension, the curve must start decreasing. Duda *et al.* [8, Subsection 9.6.7] claims that for many real-world problems they decay monotonically. Tax and Duin [121] calls it expected that performance improves with more data and Gu *et al.* [36] makes a similar claim, Meng and Xie [122] and Ting *et al.* [123] consider it conventional wisdom, and Boonyanunta and Zeephongsekul [95] considers it widely accepted. Others assume well-behaved curves [24], which usually means that curves are smooth and monotone [124]. Amari *et al.* [115] states that the generalization error decreases as training set size increases. Further note that most works in Subsection 4.4.2 only consider monotone parametric models.

Meanwhile, the peaking phenomena, where the learning curve has a local maximum, was already discovered in 1989 [125]. Such non-monotone behaviors of the learning curve have only recently gained widespread attention in the machine learning community since the publication of Belkin *et al.* [52] which covers a closely related phenomena called Double Descent, which is actually a phenomem of feature curves. Our hypothesis is that these statements are made in literature because the works illustrating non-monotone behavior have received little attention in the machine learning community. Besides peaking, there are several more distinct phenomena of non-monotone learning curves. Figures 4.4a to 4.4g provides an overview of types of ill-behaved learning curve shapes in the order we will discuss them in this section. For each we discuss potential causes (if known) and possible remedies. The code reproducing these curves (all based on actual experiments) can be retrieved from `https://github.com/tomviering/ill-behaved-learning-curves`. Standard errors are small because of many runs (≥50) and therefore are not displayed.

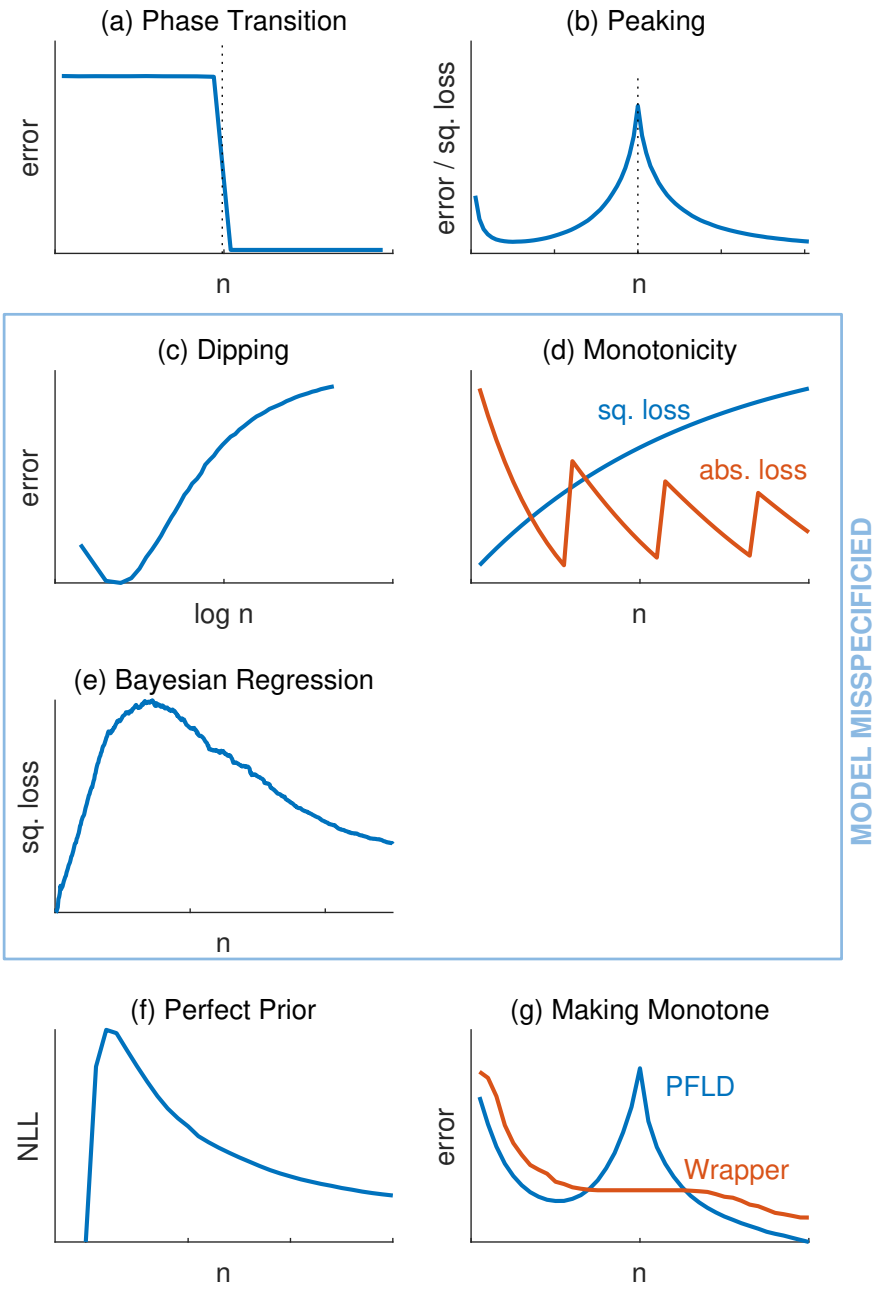


Figure 4.4.: Qualitative overview of various learning curve shapes placed in different categories with references to their corresponding subsections. All have the sample size n or $\log n$ on the horizontal axis. Dotted lines indicate the transition from under to overparametrized models. Abbreviations; error: error rate; sq. loss: squared loss; NLL: negative log likelihood; abs. loss: absolute loss; PA indicates the problem-average learning curve is shown. The 'Wrapper' algorithm aims to make learning curves of other algorithms, such as the Pseudo-Fisher, more monotone.

Before this section addresses actual bad behavior, we cover phase transitions, which are at the brink of becoming ill-behaved. In the last subsection we discuss general approaches to make the learning curve monotone, i.e. approaches that make the curve monotone irrespective of the underlying cause.

4.6.1. PHASE TRANSITIONS

As for physical systems, in a phase transition, particular learning curve properties change relatively abruptly, (almost) discontinuously. Figure 4.4a gives an example of how this can manifest itself. In learning, techniques from statistical physics can be employed to model and analyze these transitions, where it typically is studied in the limit of large samples and high input dimensionality [61]. The name phase transition of the learning curve comes from the fact that the statistical physics model undergoes a physical phase transition where continuous quantities do show actual discontinuous jumps. Most theoretical insights are limited to relatively simple learners, like the perceptron, and often apply to PA curves.

Let us point out that abrupt changes also seem to occur in human learning curves [126, 127], in particular when the task is complex and has a hierarchical structure [128]. A first mention of the occurrence of phase transitions, explicitly in the context of learning curves, was by Patarnello and Carnevali [129]. It indicates the transition from memorization to generalization, which occurs, roughly, around the time that the full capacity of the learner has been used. Györgyi [130] provides a first, more rigorous demonstration within the framework of statistical physics—notably, for the approximation referred to as the thermodynamic limit [61]. Transitions happen for single-layer perceptrons where weights take on binary values only.

The perceptron and its thermodynamic limit are considered in many later studies as well. The general finding is that, when using discrete parameter values—most often binary weights, phase transitions can occur [117, 131, 132]. The behavior is often characterized by long plateaus where the perceptron cannot learn at all (usually in the overparametrized, memorization phase, where $n < d$) and has random guessing performance, until a point where the perceptron starts to learn (at $n > d$, the underparametrized regime) at which a discontinuous jump occurs to non-trivial performance. Note that we have covered that discrete features in some empirical studies seem to lead to exponential curves [35, 96] (in Section 4.4.2), the relation with these more theoretical results regarding phase transitions remains unclear and has not yet been further explored to our knowledge.

Phase transitions are also found in two-layer networks with binary weights and activations [132–134]. This happens for the parity problem where the aim is to detect the parity of a binary string [135] for which Opper [136] found phase transitions in approximations of the learning curve. Learning curve bounds derived with statistical physics techniques may display phase transitions as well [106, 137], though Seung and Sompolinsky [117, 137] question whether these will also occur in the actual learning curve. Note that learning curve bounds are not easily (and rarely) computed for practical settings (often relying on unknown constants). Both Sompolinsky [138] and Opper [136] note that the sharp phase transitions predicted by theory, will be more gradual in real-world settings. Indeed, when studying this literature, one should be careful in interpreting the results. If one wants to run an actual experiment, it can be difficult to find the phase transitions. In case of binary features, learning can be intractable or very time consuming to simulate. Furthermore,

many of the theoretical works often study the learning curve under limiting cases such as the thermodynamical limit, where the number of training samples and dimensionality go to infinity. It remains unclear if the learning curve for small sample sizes and dimensionality shows phase transitions in this case.

For unsupervised learning, phase transitions have been shown to occur [139, 140] (see the latter for additional references). Ipsen and Hansen [141] extend these analyses to PCA with missing data. They also show phase transitions in experiments on real-world data sets. Bhat *et al.* [142] provides one of the few real application papers where a distinct, intermediate plateau is visible in the curve.

For Figure 4.4a, we constructed a simple yet novel phase transition based on a two-class classification problem, $y \in \{+1, -1\}$, with the first $d-1$ features standard normal and the dth feature set to $\frac{y}{d}$. Pseudo-Fisher's performance shows a transition at $n = d$ for the error rate, which is particularly sharp for $d \geq 4$, irrespective of whether stratified or non-stratified sampling is used. The transition occurs because if $n < d$, Pseudo-Fisher is unlikely to find the right separating hyperplane because all the 'wrong' features have a higher variance than the 'right' one. While if $n \geq d$ it is almost certain the right hyperplane is found.

4.6.2. PEAKING AND DOUBLE DESCENT

The term peaking indicates that the learning curve takes on a maximum, typically in the form of a cusp, around the point where $n \approx d$, see Figure 4.4b. This learning curve was generated by taking a horizontal slice of the surfaceplot of Figure 4.2, see the caption for a description of the problem. The training set was collected in a stratified manner, but we note that stratified versus non-stratified makes little difference for the learning curve.

Unlike many other ill behaviors, peaking can occur in the realizable setting. Its cause seems related to instability of the model. This peaking should not be confused with peaking for feature curves as covered in Subsection 4.2.7, which is related to the curse of dimensionality. Nevertheless, the same instability that causes peaking in learning curves can also lead to a peak in feature curves, see Figure 4.2. The latter phenomenon has gained quite some renewed attention in recent years under the name double descent [52].

By now, (sample-wise) double descent has become a term for the peak in the learning curve for deep neural networks [56, 143]. Related terminologies are model-wise double descent, that describe a peak in the plot of performance versus model size, and epoch-wise double descent, that shows a peak in the training curve [56].

Peaking was first observed for the Pseudo-Fisher classifier [49] and has been studied already for quite some time [125]. Pseudo-Fisher often peaks at $d \approx n$, both for the squared loss and classification error. A first theoretical model explaining this behavior in the thermodynamical limit is given in [59]. In such works, originating from statistical physics, the usual quantity of interest is $\alpha = \frac{d}{n}$ that controls the relative sizes for d and n going to infinity [60, 61, 114].

Raudys and Duin [73] investigate this behavior in the finite sample setting where each class is a Gaussian. They approximately decompose the generalization error in three terms. The first term measures the quality of the estimated means and the second the effect of reducing the dimensionality due to the pseudo-inverse. These terms reduce the error when n increases. The third term measures the quality of the estimated eigenvalues of the covariance matrix. This term increases the error when n increases, because more eigenvalues need

to be estimated at the same time if n grows, reducing the quality of their overall estimation. These eigenvalues are often small and as the model depends on their inverse, small estimation errors can have a large effect, leading to a large instability [144] and this also explains the peak in the learning curve around $n \approx d$. Using an analysis similar to that of Raudys and Duin [73], Krijthe and Loog [145] studies the peaking phenomenon in semi-supervised learning ([10]) and shows that unlabeled data can both mitigate or worsen it.

Peaking of the Pseudo-Fisher can be avoided through regularization, e.g., by adding λI to the covariance matrix [73, 144]. The performance of the model is, however, very sensitive to the correct tuning of the ridge parameter λ [144, 146]. Assuming the features are isotropic and a well-specified linear regression model is used, Nakkiran *et al.* [147] proves that peaking disappears for the optimal setting of the regularization parameter.

Other, more heuristic solutions change the training procedure altogether, e.g., Duin [148] uses an iterative procedure that decides which objects Pseudo-Fisher should be trained on, as such reducing the size of the training set, and this way the peak is avoided. Skurichina and Duin [149] adds copies of objects with noise, increasing n, or increases the dimensionality by adding noise features, increasing d. By artificially increasing n, one can bypass the peak, or by changing d, the peak can be moved to another location in the learning curve to avoid it. Their experiments show this can indeed avoid peaking, but one may wonder if the problem is 'solved' by these approaches.

Duin [50] illustrates experimentally that the SVM may not suffer from peaking in the first place. Opper [136] suggests a similar conclusion based on a thought experiment. For specific learning problems, both Opper *et al.* [59] and Watkin *et al.* [131] already give a theoretical underpinning for the absence of double descent for the perceptron of optimal (or maximal) stability, which is a classifier closely related to the SVM. Opper and Urbanczik [150] studies the behavior of the SVM in the thermodynamic limit which does not show peaking either. Spigler *et al.* [151] show, however, that double descent for feature curves can occur using the (squared) hinge loss, where the peak is typically located at an $n > d$.

Further insight of when peaking can occur may be gained from recent works by Advani and Saxe [152] and Hastie *et al.* [153], which perform a rigorous analysis of the case of Fourier Features with Pseudo-Fisher using random matrix theory. Advani and Saxe [152] studies the dynamics of gradient descent for deep learning, and uses Pseudo-Fisher as a surrogate model for deepnets. He finds several explanations why overfitting is not an issue for such deep networks. Hastie *et al.* [153] analyses the case where features are i.i.d. and transformed by a linear transformation, and where features are processed by a random neural network of one layer. Results should, however, be interpreted with care as these are typically derived in an asymptotic setting where both n and d (or some more appropriate measure of complexity) go to infinity, i.e., a setting similar to the earlier mentioned thermodynamic limit.

Furthermore, d'Ascoli *et al.* [154] shows that a peak can occur where the training set size n equals the input dimensionality d, but also when n matches the number of parameters of the learner, depending on the latter's degree of nonlinearity. They identify that there are two peaks, one related to overfitting noise in the labels, and one peak due to sensitivity of the learner and its random initialisation. Multiple peaks are also possible for $n < d$ [147], by creating block structures in the covariance matrix of the features.

4.6.3. DIPPING AND OBJECTIVE MISMATCH

In dipping, the learning curve may initially improve with more samples, but the performance eventually deteriorates and never recovers, even in the limit [155], see Figure 4.4c. A 1D toy problem for which many well-known linear classifiers (e.g., SVM, logistic regression, LDA, Pseudo-Fisher, nearest-mean) dip is given in Figure 4.5. Training nearest-mean, a classifier that assigns object to the class that has the nearest empirical mean on the training set, on this distribution results in the learning curve in Figure 4.4c. Note that we used non-stratified sampling.

Let us describe why this happens for the example distribution in the figure. If we have only a training set of two samples, we will in 50% of the cases obtain the optimal model with an error rate of 25% (shown as dotted lines). In the other 50% of the cases, we will have 2 samples from the same class (unless sampling is stratified), which results in a model that classifies everything as that observed class, which thus has an error rate of 50%. Thus the expected error rate for $n = 2$ is 37.5% in the case of non-stratified sampling, and 25% in the case of stratified sampling (because in the latter case, we will have one sample for each class). However, in the limit, a model is obtained with a decision boundary at $x = 0$, because the empirical means for that case will coincide. This model achieves an error rate of 50%. Thus the best expected performance is reached for a finite training set size (of around 2 samples). What is essential for dipping to occur is that the classification problem at hand is misspecified, and that the learner optimizes something else than the evaluation metric of the learning curve. Such *objective misspecification* is standard since many evaluation measures such as error rate, AUC, F-measure, and so on, are notoriously hard to optimize. To illustrate, minimizing the error rate for linear classifiers is NP-hard in the agnostic case when n and d are increased and also NP-hard to approximate within a constant factor [102, page 105-119]. If classification-calibrated loss functions are used and the hypothesis class is rich enough to contain the true model, then minimizing the surrogate loss will also minimize the error rate [156, 157]. Thus a more complex hypothesis class may fix the dipping problem.

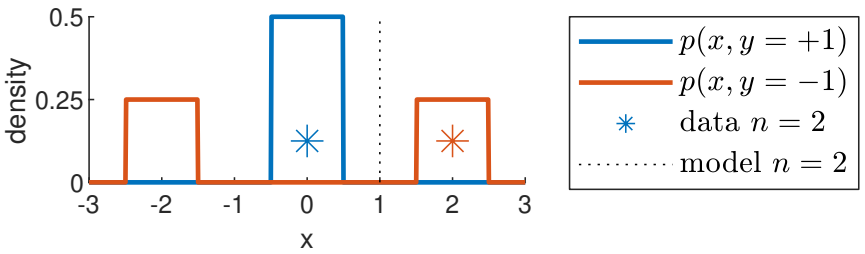


Figure 4.5.: A one dimensional two-class problem that causes dipping of the learning curve for various linear classifiers [155]. The sample data (the two $*$s) illustrates that, with small samples, the optimal linear model in terms of error rate can be obtained. However, due to the surrogate loss many classifiers optimize, the decision boundary they find in the limit of infinite sample sizes will be around $x = 0$, which is suboptimal.

By constructing an explicit problem Devroye *et al.* [2, page 106] already showed that the nearest neighbor classifier is not always smart, meaning its learning curve can go up locally. A similar claim is made for kernel rules [2, Problems 6.14 and 6.15]. In a different context, Ben-david *et al.* [157] provide an even stronger example where all linear classifiers optimizing a convex surrogate loss converge in the limit to the worst possible classifier for which the error rate approaches 1. Another example, Lemma 15.1 in [2], gives a case of dipping for likelihood estimation.

Other works also show dipping of some sort. For example, Frey and Fisher [32] fit C4.5 to a synthetic dataset that has binary features for which the parity of all features determines the label. When fitting C4.5 the test error increases with the amount of training samples. They attribute this to the fact that the C4.5 is using a greedy approach to minimize the error, and thus is closely related to objective misspecification. Brumen *et al.* [92] also shows an ill-behaving curve of C4.5 that seems to go up. They note that 34 more curves could not be fitted well using their parametric models, where possibly something similar is going on. In the work of Vanschoren *et al.* [158] we can find another potential example of dipping as, in Figure 6, the accuracy goes down with increasing sample sizes.

Anomaly or outlier detection using k-nearest neighbors (kNN) can also show dipping behavior [123] (referred to as gravity-defying learning curves). Also here is a mismatch between the objective that is evaluated with, i.e., the AUC, and kNN that does not optimize the AUC. Hess and Wei [19] also show kNN learning curves that deteriorate in terms of AUC in the standard supervised setting.

Also in active learning [11] for classification, where the test error rate is often plotted against the size of the (actively sampled) training set, learning curves are regularly reported to dip [159, 160]. In that case, active learners provide optimal performance for a number of labeled samples that is smaller than the complete training set. This could be interpreted as a great success for active learning. It implies that even in regular supervised learning, one should maybe use an active learner to pick a subset from one's complete training set, as this can improve performance. It cannot be ruled out, therefore, that the active learner uses an objective that matches better with the evaluation measure [161].

Meng and Xie [122] construct a dipping curve in the context of time series modeling with ordinary least squares. In their setting, they use an adequate parametric model, but the distribution of the noise changes every time step, which leads least squares to dipping. In this case, using the correct likelihood to fit the model resolves the non-monotonicity.

Finally, negative transfer [162] in transfer learning and domain adaptation [163, 164], can be interpreted as dipping as well. In this case, more source data deteriorates performance on the target and the objective mismatch stems from the combined training from source and target data instead of the latter only.

4.6.4. RISK MONOTONICITY AND ERM

Several novel examples of non-monotonic behavior for density estimation, classification, and regression by means of standard empirical risk minimization (ERM) are presented by Loog *et al.* [165]. Similar to dipping, the squared loss increases with n, but in contrast does eventually recover, see Figure 4.4d. However, these examples cannot be explained either in terms of dipping or peaking. Dipping is ruled out as, in ERM, the learner optimizes the loss that is used for evaluation. In addition, non-monotonicity can be demonstrated for any

n and so there is no direct link with the capacity of the learner, ruling out an explanation in terms of peaking.

Proofs of non-monotonicity are given for squared, absolute, and hinge loss. It is demonstrated that likelihood estimators suffer the same deficiency. Two learners are reported that are provably monotonic: mean estimation based on the squared loss and the memorize algorithm. The latter algorithm does not really learn but outputs the majority voted classification label of each object if it has been seen before. Memorize is not PAC learnable [61, 102], illustrating that monotonicity and PAC are essentially different concepts. It is shown experimentally that regularization can actually worsen the non-monotonic behavior. Why regularization may lead to non-monotonicity is explained in more detail in Section 4.6.6. In contrast, Nakkiran *et al.* [147] shows that optimal tuning of the regularization parameter can guarantee monotonicity in certain settings. A final experiment by Loog *et al.* [165] shows a surprisingly jagged learning curve for the absolute loss, see Figure 4.4.

4.6.5. MISSPECIFIED BAYESIAN REGRESSION AND SAFEBAYES

Grünwald and van Ommen [166] show that a (hierarchical) Bayesian linear regression model can give a broad peak in the learning curve of the squared risk, see Figure 4.4e. One way this can happen is when the homogeneous noise assumption is violated, while the estimator is otherwise consistent.

Specifically, let data be generated as follows. For each sample, a fair coin is flipped. Heads means the sample is generated according to the ground truth probabilistic model contained in the hypothesis class, $y = w^T x + \epsilon$ (with no bias term), where ϵ is zero mean Gaussian noise with fixed variance. Misspecification happens when the coin comes up tails and a sample with $x = 0$ and $y = 0$ is returned. The peak in the learning curve cannot be explained by dipping (because no surrogate loss is used) or by the same underlying principle as peaking, as the peak location is not dependent on the dimensionality. Furthermore, it does not occur because of the known sensitivity of the squared loss to outliers according to Grünwald and Van Ommen. This is because the sample is an inlier: it is located at $(0,0)$, and therefore it can be perfectly predicted by the linear model. Thus this is truly some surprising behavior of Bayesian linear regression.

The peak in the learning curve is fairly broad and occurs in various experiments. As also no approximations are to blame, the authors conclude that Bayes' rule is really at fault as it cannot handle the misspecification. The reasoning why things go wrong is fairly technical and we do not go into it here. However, we can explain one necessary condition: non-monotonicity can happen if the probabilistic model class is not convex. The Bayesian linear regression model assumes

$$p(y|x,w,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp -\frac{1}{2}\left(\frac{w^T x}{\sigma}\right)^2. \tag{4.10}$$

Now if we take a convex combination of this likelihood model, such as $\frac{1}{2}p(y|x,w_1,\sigma) + \frac{1}{2}p(y|x,w_2,\sigma)$, it is not part of the original model class anymore. Because in that case the convex combination is bimodal instead of unimodal, and thus the model class is non-convex in this sense.

After the in depth analysis, Grünwald and Van Ommen introduce a modified Bayes rule in which the likelihood is raised to some power η. The parameter η cannot be learned in a Bayesian way, leading to their SafeBayes approach. Their technique alleviates the broad peak in the learning curve and is empirically shown to make the curves generally more well-behaved.

4.6.6. THE PERFECT PRIOR

As we have seen in Subsection 4.5.3, the PA learning curve is always monotone if the problem is well specified and a Bayesian decision theoretical approach is followed. Nonetheless, the fact that the PA curve is monotone does not mean that the curve for every individual problem is. Grünwald and Kotłowski [167] offers an insightful example (see also Figure 4.4f): consider a fair coin and let us estimate its probability p of heads using Bayes' rule. We measure performance using the negative log-likelihood on an unseen coin flip and adopt a uniform Beta(1,1) prior on p. This prior, i.e., without any training samples, already achieves the optimal loss since it assigns the same probability to heads and tails. After a single flip, $n = 1$, the posterior is updated and leads to a probabilities of $\frac{1}{3}$ or $\frac{2}{3}$ and the loss *must* increase. Eventually, with $n \to \infty$, the optimal loss is recovered, forming a bump in the learning curve. Note that this construction is rather versatile and can create non-monotonic behavior for practically any Bayesian estimation task. In a similar way, any type of regularization can lead to comparable learning curve shapes (see also Subsection 4.6.4).

A related example is given by Al-Saleh and Masoud [168]. They show that the posterior variance can also increase for a single problem, unless the likelihood belongs to the exponential family and a conjugate prior is used.

4.6.7. MONOTONICITY: A GENERAL FIX?

This section has noted a few particular approaches to restore monotonicity of a learning curve. One may wonder, however, whether generally applicable approaches exist that can turn any learner into a monotone one. A first attempt is made by Viering *et al.* [169] (Chapter 6). They propose a wrapper that, with high probability, makes any classifier monotone in terms of the error rate. The main idea is to consider n as a variable over which model selection is performed. When n is increased, a model trained with more data is compared to the previously best model on validation data. Only if the new model is judged to be significantly better—following a hypothesis test, the older model is discarded. If the original learning algorithm is consistent and if the size of the validation data grows, the resulting algorithm is consistent as well. It is empirically observed that the monotone version may learn more slowly, giving rise to the question whether there always will be a trade-off between monotonicity and speed (refer to the learning curve in Figure 4.4g).

Recently, Mhammedi and Husain [170] extended this idea, proposing two algorithms that do not need to set aside validation data while guaranteeing monotonicity. To this end they assume that the Rademacher complexity of the hypothesis class composited with the loss is finite. The Rademacher complexity measures the complexity of the hypothesis class, similar to the VC dimension (see [102, Chapter 26]). This allows them to determine when to switch to a model trained with more data. In contrast to Viering *et al.* [169], Mhammedi and Husain [170] argue that their second algorithm does not learn slower, as the rate of

convergence of the base learner remains the same up to constants.

4.7. DISCUSSION

We have covered various examples that illustrate that learning curves can be quite ill-behaved, which further emphasizes what little we know about learning curve behavior. It is evident that there is no theory that covers all the different characteristics of learning curves that we have covered.

The usage of the learning curve as a tool for the analysis of learning algorithms has varied throughout the past decades. In line with Langley, Perlich, Hoiem, et al. [23, 28, 30], we would like to suggest a more consistent use. We specifically agree with Perlich *et al.* [23] that without a study of the learning curves, claims of superiority of one approach over another are perhaps only valid for very particular sample sizes. Reporting learning curves in empirical studies can also help the field to move away from its fixation on bold numbers, besides accelerating learning curve research. To that latter end, the openly accessible learning curve database recently published [40] provides ample opportunity for a collaborative investigation.

In the years to come, we expect investigations of parametric models and their performance in terms of extrapolation. Insights into these problems become more and more important—particularly within the context of deep learning—to enable the intelligent use of computational resources. In the remainder, we highlight some specific aspects that we see as important.

4.7.1. AVERAGING CURVES AND THE IDEAL PARAMETRIC MODEL

Especially for extrapolation, a learning curve should be predictable, which, in turn, asks for a good parametric model. It seems impossible to find a generally applicable, parametric model that covers all aspects of the shape, in particular ill-behaving curves. Nevertheless, we can try to find a model class that would give us sufficient flexibility and extrapolative power. Power laws and exponentials should probably be part of that class, but does that suffice?

Recent work has indicated that the 4-parameter models also have potential — are these models simply approximating power laws or exponentials, or do they truly have additional modeling power that is necessary to capture learning curve behavior? Furthermore, some parametric models have particular invariants (such as invariance to rescaling); can our prior knowledge of the risk guide us to which invariants a parametric model should have for modeling learning curves?

To get close to the true learning curve, some studies average hundreds or thousands of individual learning curves [13, 15]. Averaging can mask interesting characteristics of individual curves [171, 172]. This has been extensively debated in psychonomics, where cases have been made for exponentially shaped individual curves, but power-law-like average curves [173, 174]. In applications, we may need to be able to model individual, single-training-set curves or curves that are formed on the basis of relatively small samples. As such, we see potential in studying and fitting individual curves to better understanding their behavior, which are potentially more difficult to fit.

4.7.2. HOW TO ROBUSTLY FIT LEARNING CURVES

A technical problem that has received little attention is how to properly fit a parametric model to a learning curve. As far as current studies at all mention how the fitting is carried out, many seem to rely on simple least squares fitting. Two recent studies indeed indicate that indeed many fits fail (as many as 2%), and that curve fitting often fails to beat a simple baseline that always predicts the last observed risk. This illustrates there seems much room to further improve curve fitting for learning curves.

A probabilistic model with assumptions that more closely match the learning curve seems promising to investigate. Given the tricky nature of extrapolation, particularly when dealing with relatively small number of points for fitting the learning curve, further investigating robust estimation methods that match well with the intended purpose should also be worthwhile. Here, we see potential in combining parametric and non-parametric techniques for curve extrapolation, possibly in combination with meta-learning.

4.7.3. BOUNDS AND ALTERNATIVE STATISTICS

One should be careful in interpreting theoretical results when it comes to the shape of learning curves. Generalization bounds, such as those provided by PAC, that hold uniformly over all P may not correctly characterize the curve shape. Similarly, strictly decreasing bounds do not imply monotone learning curves, and thus do not rule out ill-behavior. Furthermore, PA learning curves, which require an additional average over problems, can show behavior substantially different from those for a single problem, because here averaging can also mask characteristics.

Another incompatibility between typical generalization bounds and learning curves is that the former are constructed to hold with *high probability* with respect to the sampling of the training set, while the latter look at the *mean* performance over all training sets. Though bounds of one type can be transformed into the other [175], this conversion can change the actual shape of the bound, thus such high probability bounds may also not correctly characterize learning curve shape.

The preceding can also be a motivation to actually study learning curves for statistics other than the average. For example, in Equation 4.2, instead of the expectation we could look at the curves of the median or other percentiles. These quantities are closer related to high probability learning bounds. Of course, we would not have to choose the one learning curve over the other. They offer different types of information and, depending on our goal, may be worthwhile to study next to each other. Along the same line of thought, we could include quartiles in our plots, rather than the common error bars based on the standard deviation. Ultimately, we could even try to visualize the full loss distribution at every sample size n and, potentially, uncover behavior much more rich and unexpected.

A final estimate that we think should be investigated more extensively is the training loss. Not only can this quantity aid in identifying overfitting and underfitting issues [48, 172], but it is a quantity that is interesting to study as such or, say, in combination with the true risk. Their difference, a simple measure of overfitting, could, for example, turn out to behave more regular than the two individual measures (as also suggested by Cortes *et al.* [31]).

4.7.4. RESEARCH INTO ILL-BEHAVIOR AND META-LEARNING

We believe better understanding is needed regarding the occurrence of peaking, dipping, and otherwise non-monotonic or phase-transition-like behavior: when and why does this happen? Certainly, a sufficiently valid reason to investigate these phenomena is to quench one's scientific curiosity. We should also be careful, however, not to mindlessly dismiss such behavior as mere oddity. Granted, these uncommon and unexpected learning curves have often been demonstrated in artificial or simplistic settings, but this is done to make at all insightful that there is a problem.

The simple fact is that, at this point, we do not know what role these phenomena play in real-world problems. Now that many benchmark datasets are readily available, this issue can be studied more rigorously. A first step towards that end is provided by the openly accessible learning curve database of Mohr *et al.* [40], who do find some preliminary evidence that non-monotone curves seem to be rare, but we believe this warrants further investigation. Properly summarizing (see Section 4.2.4) and openly sharing learning curve data via this database can further accelerate this research. Automated techniques may then be developed to find curious learning curve phenomena and possibly predict them.

Given the success of meta-learning for curve extrapolation and model selection this seems a promising possibility. Such meta-learning studies on large amounts of datasets could, in addition, shed more light on what determines the parameters of learning curve models, a topic that has been investigated relatively little up to now. Predicting these parameters robustly from very few points along the learning curve will prove valuable for virtually all applications.

4.7.5. OPEN THEORETICAL QUESTIONS

There are two rather specific theoretical questions that we would like to draw attention to. Both are concerned with the monotonicity of a learner.

The first one asks whether maximum likelihood estimators for well-specified models behave monotonically. Likelihood estimation, being a century-old, classical technique [176, 177], has been heavily studied, both theoretically and empirically. In much of the theory developed, the assumption that one is dealing with a correctly specified model is common, but we are not aware of any results that demonstrate that better models are obtained with more data. The question is interesting for the likelihood exactly because this estimator has been extensively studied already and still plays a central role in statistics and abutting fields.

The second question is broader: for standard classification and regression problems, among the consistent learners are there monotonic ones? We saw that we could make them more monotonic in some settings, but the question is whether making them strictly monotonic is possible as well. Is there a general solution? Interestingly, specifically for universally consistent classification rules, Devroye *et al.* [2, page 106] conjecture that this is not possible.

4.8. CONCLUSION

It should be clear that learning curves have a lot of potential: speeding up learning, tuning of hyperparameters, improving model selection, and estimating the amount of data required to reach a certain performance. However, to really make the most of learning curves, we need a better understanding of their shape.

It is apparent that there is still a lot we do not know about learning curves and their shapes. Various empirical studies for shallow learners show contradicting results, and give some indication no universal parametric model may be identified. Recent studies indicate that the power law with 3 parameter or 4-parameter models may perform best overall compared to the other parametric models in Table 4.1. For deep learning, the evidence for the power law is more convincing, but sufficient tuning of hyperparameters is necessary, otherwise learning is slower.

There are some theoretical works that favor a power law or exponential learning curve. However, there is a large gap between the theoretical works and practice, such as various strong assumptions or unusual learners. While some theoretical results do corroborate some empirical findings, one should perhaps treat this as a mere coincidence. The recent line of work of Bousquet *et al.* [108] seems most promising to further develop in order to understand curve shapes better. Finally, older statistical physics works (like those of Amari) may also give plenty of inspiration how to analyse modern deep neural networks.

We have illustrated with various learning problems that learning curves can be ill-behaved. It should be apparent that these are not artefacts, but are actual learning problems where curves really are ill-behaved. For some of them, specific remedies have been discussed, but most actually emphasize how little we understand about learning curves. A recent line of work aims to develop wrapper algorithms to make any learner, irrespective of the underlying causes for the non-mononitinicty, better behaved. This is a promising opion to turn any learner into a monotone one.

In the foregoing, we identified some specific challenges already, but we are convinced that many more open and interesting problems can be discovered. In this, the current review should help both as a guide and as a reference.

4.9. BIBLIOGRAPHY

[1] T. Viering and M. Loog, *The shape of learning curves: a review,* arXiv preprint arXiv:2103.10948 (2021).

[2] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Vol. 31 (Springer, New York, NY, USA, 1996).

[3] T. Viering, A. Mey, and M. Loog, *Open problem: Monotonicity of learning,* in *Conference on Learning Theory* (2019) pp. 3198–3201.

[4] C. Perlich, *Learning curves in machine learning,* in *Encyclopedia of Machine Learning*, edited by C. Sammut and G. I. Webb (Springer, 2010) pp. 577–488.

[5] C. Sammut and G. I. Webb, *Encyclopedia of machine learning* (Springer Science & Business Media, 2011).

[6] L. E. Atlas, D. A. Cohn, and R. E. Ladner, *Training connectionist networks with queries and selective sampling,* in *NeurIPS* (1990) pp. 566–573.

[7] H. Sompolinsky, N. Tishby, and H. S. Seung, *Learning from examples in large neural networks,* Physical Review Letters **65**, 1683 (1990).

[8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification* (John Wiley & Sons, 2012).

[9] K. P. Murphy, *Machine learning: a probabilistic perspective* (MIT press, 2012).

[10] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning* (The MIT Press, 2010).

[11] B. Settles, *Active learning literature survey*, Tech. Rep. (University of Wisconsin-Madison, Department of Computer Sciences, 2009).

[12] S. Mukherjee, P. Tamayo, S. Rogers, R. Rifkin, A. Engle, C. Campbell, T. R. Golub, and J. P. Mesirov, *Estimating dataset size requirements for classifying dna microarray data,* Journal of computational biology **10**, 119 (2003).

[13] R. L. Figueroa, Q. Zeng-Treitler, S. Kandula, and L. H. Ngo, *Predicting sample size required for classification performance,* BMC med. inf. and decision making **12**, 8 (2012).

[14] A. N. Richter and T. M. Khoshgoftaar, *Learning curve estimation with large imbalanced datasets,* in *ICMLA* (2019) pp. 763–768.

[15] A. Y. Ng and M. I. Jordan, *On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes,* in *NeurIPS* (2002) pp. 841–848.

[16] `https://waikato.github.io/weka-wiki/experimenter/learning_curves/`.

[17] `https://github.com/scikit-learn/scikit-learn/blob/0fb307bf3/sklearn/model_selection/_validation.py#L1105`.

[18] J.-H. Kim, *Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap,* Computational statistics & data analysis **53**, 3735 (2009).

[19] K. R. Hess and C. Wei, *Learning curves in classification with microarray data,* in *Seminars in oncology*, Vol. 37 (Elsevier, 2010) pp. 65–68.

[20] `http://prtools.tudelft.nl/`.

[21] B. Efron, *Estimating the error rate of a prediction rule: improvement on cross-validation,* Journal of the American statistical association **78**, 316 (1983).

[22] A. K. Jain, R. C. Dubes, and C.-C. Chen, *Bootstrap techniques for error estimation,* TPAMI , 628 (1987).

[23] C. Perlich, F. Provost, and J. S. Simonoff, *Tree Induction vs. Logistic Regression: A Learning-Curve Analysis,* JMLR **4**, 211 (2003).

[24] F. Provost, D. Jensen, and T. Oates, *Efficient progressive sampling,* in *ACM SIGKDD* (1999) pp. 23–32.

[25] E. Perez and L. A. Rendell, *Using multidimensional projection to find relations,* in *ML Proc.* (Elsevier, 1995) pp. 447–455.

[26] D. Mazzoni and K. Wagstaff, *Active learning in the presence of unlabelable examples*, Tech. Rep. (NASA/JPL, 2004).

[27] B. Settles and M. Craven, *An analysis of active learning strategies for sequence labeling tasks,* in *Conference on Empirical Methods in Natural Language Processing* (2008) pp. 1070–1079.

[28] P. Langley, *Machine Learning as an Experimental Science,* Machine Learning **3**, 5 (1988).

[29] N. Bertoldi, M. Cettolo, M. Federico, and B. Christian, *Evaluating the learning curve of domain adaptive statistical machine translation systems,* in *Workshop on Statistical Machine Translation* (2012) pp. 433–441.

[30] D. Hoiem, T. Gupta, Z. Li, and M. M. Shlapentokh-Rothman, *Learning curves for analysis of deep networks,* (2020), arXiv:2010.11029 [cs.LG] .

[31] C. Cortes, L. D. Jackel, S. A. Solla, V. Vapnik, and J. S. Denker, *Learning curves: Asymptotic values and rate of convergence,* in *NeurIPS* (1994) pp. 327–334.

[32] L. J. Frey and D. H. Fisher, *Modeling decision tree performance with the power law.* in *AISTATS* (1999).

[33] S. Singh, *Modeling performance of different classification methods: deviation from the power law,* Project Report, Department of Computer Science, Vanderbilt University, USA (2005).

[34] M. Last, *Predicting and optimizing classifier utility with the power law,* in *ICDMW* (IEEE, 2007) pp. 219–224.

[35] D. Cohn and G. Tesauro, *Can neural networks do better than the vapnik-chervonenkis bounds?* in *NeurIPS* (1991) pp. 911–917.

[36] B. Gu, F. Hu, and H. Liu, *Modelling classification performance for large data sets,* in *International Conference on Web-Age Information Management* (Springer, 2001) pp. 317–328.

[37] P. Kolachina, N. Cancedda, M. Dymetman, and S. Venkatapathy, *Prediction of Learning Curves in Machine Translation,* in *ACL* (Jeju Island, Korea, 2012) pp. 22–30.

[38] A.-N. Spiess and N. Neumeyer, *An evaluation of r2 as an inadequate measure for nonlinear models in pharmacological and biochemical research: a monte carlo approach,* BMC pharmacology **10**, 1 (2010).

[39] B. Brumen, A. Černezel, and L. Bošnjak, *Overview of machine learning process modelling,* Entropy **23**, 1123 (2021).

[40] F. Mohr, T. J. Viering, M. Loog, and J. N. van Rijn, *LCDB 1.0: An extensive learning curves database for classification tasks,* in *Machine Learning and Knowledge Discovery in Databases, ECMLPKDD*, Lecture Notes in Computer Science (Springer, 2022) p. accepted.

[41] R. P. Duin, *Classifiers in almost empty spaces,* in *ICPR*, Vol. 2 (IEEE, 2000) pp. 1–7.

[42] C. M. Bishop, *Pattern recognition and machine learning* (springer, 2006).

[43] G. Hughes, *On the mean accuracy of statistical pattern recognizers,* IEEE Trans. IT **14**, 55 (1968).

[44] A. Jain and B. Chandrasekaran, *Dimensionality and Sample Size Considerations in Pattern Recognition Practice,* Handbook of Statistics **2**, 835 (1982).

[45] J. M. Van Campenhout, *On the peaking of the hughes mean recognition accuracy: the resolution of an apparent paradox,* IEEE Transactions on SMC **8**, 390 (1978).

[46] A. K. Jain and B. Chandrasekaran, *Dimensionality and sample size considerations in pattern recognition practice,* in *Handbook of statistics*, Vol. 2, edited by P. Krishnaiah and L. Kanal (Elsevier, 1982) Chap. 39, pp. 835–855.

[47] S. Raudys and V. Pikelis, *On dimensionality, sample size, classification error, and complexity of classification algorithm in pattern recognition,* TPAMI , 242 (1980).

[48] A. K. Jain, R. P. W. Duin, and J. Mao, *Statistical pattern recognition: A review,* TPAMI **22**, 4 (2000).

[49] F. Vallet, J.-G. Cailton, and P. Refregier, *Linear and nonlinear extension of the pseudo-inverse solution for learning boolean functions,* EPL (Europhysics Letters) **9**, 315 (1989).

[50] R. P. Duin, *Classifiers in almost empty spaces,* in *ICPR*, Vol. 15 (2000) pp. 1–7.

[51] A. Zollanvari, A. P. James, and R. Sameni, *A theoretical analysis of the peaking phenomenon in classification,* Journal of Classification , 1 (2019).

[52] M. Belkin, D. Hsu, S. Ma, and S. Mandal, *Reconciling modern machine-learning practice and the classical biasvariance trade-off,* PNAS **116**, 15849 (2019), arXiv:1812.11118 .

[53] S. Raudys and A. Jain, *Small sample size effects in statistical pattern recognition: recommendations for practitioners,* TPAMI **13**, 252 (1991).

[54] R. P. Duin, D. de Ridder, and D. M. Tax, *Experiments with a featureless approach to pattern recognition,* Pattern Recognition Letters **18**, 1159 (1997).

[55] L. Torgo, *Kernel regression trees,* in *ECML* (1997) pp. 118–127.

[56] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, *Deep double descent: Where bigger models and more data hurt,* in *ICLR* (2019).

[57] R. Duin and D. Tax, *Statistical pattern recognition,* in *Handbook Of Pattern Recognition And Computer Vision*, edited by C. H. Chen and P. S. P. Wang (World Scientific, 2005) pp. 3–24.

[58] L. Chen, Y. Min, M. Belkin, and A. Karbasi, *Multiple descent: Design your own generalization curve,* arXiv preprint arXiv:2008.01036 (2020).

[59] M. Opper, W. Kinzel, J. Kleinz, and R. Nehl, *On the ability of the optimal perceptron to generalise,* Journal of Physics A: Mathematical and General **23**, L581 (1990).

[60] T. L. Watkin, A. Rau, and M. Biehl, *The statistical mechanics of learning a rule,* Rev. of Modern Physics **65**, 499 (1993).

[61] A. Engel and C. Van den Broeck, *Statistical mechanics of learning* (Cambridge University Press, 2001).

[62] A. K. Jain and W. G. Waller, *On the optimal number of features in the classification of multivariate gaussian data,* Pattern recognition **10**, 365 (1978).

[63] R. P. W. Duin, *On the accuracy of statistical pattern recognizers*, Ph.D. thesis, Technische Hogeschool Delft (1978).

[64] M. Skurichina and R. P. Duin, *Stabilizing classifiers for very small sample sizes,* in *ICPR*, Vol. 2 (IEEE, 1996) pp. 891–896.

[65] R. Duin, *On the choice of smoothing parameters for parzen estimators of probability density functions,* IEEE Transactions on Computers **25**, 1175 (1976).

[66] J. S. Rosenfeld, A. Rosenfeld, Y. Belinkov, and N. Shavit, *A constructive prediction of the generalization error across scales,* arXiv:1909.12673 (2019).

[67] `https://paperswithcode.com/sota`.

[68] D. Sculley, J. Snoek, A. Wiltschko, and A. Rahimi, *Winner's curse? on pace, progress, and empirical rigor,* in *ICLR* (2018).

[69] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition,* Proceedings of the IEEE **86**, 2278 (1998).

[70] K. Blagec, G. Dorffner, M. Moradi, and M. Samwald, *A critical analysis of metrics used for measuring progress in artificial intelligence,* arXiv:2008.02577 (2020).

[71] S. Raudys, *On the problems of sample size in pattern recognition (in Russian),* in *Proceedings of the 2nd All-Union Conference on Statistical Methods in Control Theory* (Nauka, 1970).

[72] L. Kanal and B. Chandrasekaran, *On dimensionality and sample size in statistical pattern classification,* Pattern recognition **3**, 225 (1971).

[73] S. Raudys and R. Duin, *Expected classification error of the Fisher linear classifier with pseudo-inverse covariance matrix,* Pattern Recognition Letters **19**, 385 (1998).

[74] R. Kohavi, *Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid.* in *Kdd*, Vol. 96 (1996) pp. 202–207.

[75] J. Bornschein, F. Visin, and S. Osindero, *Small data, big decisions: Model selection in the small-data regime,* arXiv:2009.12583 (2020).

[76] B. Strang, P. van der Putten, J. N. van Rijn, and F. Hutter, *Dont rule out simple models prematurely: a large scale benchmark comparing linear and non-linear classifiers in openml,* in *IDA* (Springer, 2018) pp. 303–315.

[77] N. Mørch, L. K. Hansen, S. C. Strother, C. Svarer, D. A. Rottenberg, B. Lautrup, R. Savoy, and O. B. Paulson, *Nonlinear versus linear models in functional neuroimaging: Learning curves and generalization crossover,* in *IPMI* (Springer, 1997) pp. 259–270.

[78] J. W. Shavlik, R. J. Mooney, and G. G. Towell, *Symbolic and neural learning algorithms: An experimental comparison,* Machine learning **6**, 111 (1991).

[79] P. Domingos and M. Pazzani, *On the optimality of the simple bayesian classifier under zero-one loss,* Machine learning **29**, 103 (1997).

[80] C. Harris-Jones and T. L. Haines, *Sample size and misclassification: Is more always better,* AMS Center for Advanced Technologies (1997).

[81] R. Duin and E. Pekalska, *Pattern Recognition: Introduction and Terminology* (37 Steps, 2016).

[82] M. Loog, *Supervised classification: Quite a brief overview,* in *Machine Learning Techniques for Space Weather* (Elsevier, 2018) pp. 113–145.

[83] S. Jones, S. Carley, and M. Harrison, *An introduction to power and sample size estimation,* Emergency medicine journal: EMJ **20**, 453 (2003).

[84] G. H. John and P. Langley, *Static versus dynamic sampling for data mining.* in *KDD*, Vol. 96 (1996) pp. 367–370.

[85] C. Meek, B. Thiesson, and D. Heckerman, *The learning-curve sampling method applied to model-based clustering,* JMLR **2**, 397 (2002).

[86] K. Tomanek and U. Hahn, *Approximating learning curves for active-learning-driven annotation.* in *LREC*, Vol. 8 (2008) pp. 1319–1324.

[87] R. Leite and P. Brazdil, *Improving progressive sampling via meta-learning on learning curves,* in *ECML* (Springer, 2004) pp. 250–261.

[88] R. Leite and P. Brazdil, *Predicting Relative Performance of Classifiers from Samples,* in *ICML* (Bonn, Germany, 2005) pp. 497—-503.

[89] R. Leite and P. Brazdil, *An iterative process for building learning curves and predicting relative performance of classifiers,* in *Portuguese Conference on Artificial Intelligence* (Springer, 2007) pp. 87–98.

[90] J. N. van Rijn, S. M. Abdulrahman, P. Brazdil, and J. Vanschoren, *Fast algorithm selection using learning curves,* in *LNCS*, Vol. 9385 (Springer Verlag, 2015) pp. 298–309.

[91] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou, *Deep Learning Scaling is Predictable, Empirically,* arXiv:1712.00409 (2017), arXiv:1712.00409 .

[92] B. Brumen, I. Rozman, M. Heričko, A. Černezel, and M. Hölbl, *Best-fit learning curve model for the c4. 5 algorithm,* Informatica **25**, 385 (2014).

[93] T. Domhan, J. T. Springenberg, and F. Hutter, *Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves,* in *Twenty-fourth international joint conference on artificial intelligence* (2015).

[94] M. J. Anzanello and F. S. Fogliatto, *Learning curve models and applications: Literature review and research directions,* Int. Journal of Industr. Ergonomics **41**, 573 (2011).

[95] N. Boonyanunta and P. Zeephongsekul, *Predicting the relationship between the size of training sample and the predictive power of classifiers,* in *KES* (Springer, 2004) pp. 529–535.

[96] S. Ahmad and G. Tesauro, *Study of scaling and generalization in neutral networks,* Neural Networks **1**, 3 (1988).

[97] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, *Revisiting Unreasonable Effectiveness of Data in Deep Learning Era,* in *ICCV* (2017) pp. 843–852.

[98] A. Joulin, L. Van Der Maaten, A. Jabri, and N. Vasilache, *Learning visual features from large weakly supervised data,* in *ECCV* (Springer, 2016) pp. 67–84.

[99] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, *Exploring the limits of weakly supervised pretraining,* in *ECCV* (2018) pp. 181–196.

[100] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, *Scaling laws for neural language models,* arXiv:2001.08361 (2020).

[101] V. Vapnik, *Estimation of dependences based on empirical data berlin,* (1982).

[102] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms* (Cambridge university press, 2014).

[103] W. L. Buntine, *A critique of the valiant model.* in *IJCAI* (1989) pp. 837–842.

[104] W. E. Sarrett and M. J. Pazzani, *Average case analysis of empirical and explanation-based learning algorithms*, Tech. Rep. 89-35 (Department of Information & Computer Science, University of California, Irvine, 1989).

[105] D. Haussler and M. Warmuth, *The probably approximately correct (pac) and other learning models,* in *Foundations of Knowledge Acquisition* (Springer, 1993) pp. 291–312.

[106] D. Haussler, M. Kearns, H. S. Seung, and N. Tishby, *Rigorous learning curve bounds from statistical mechanics [longer version],* Machine Learning **25**, 195 (1996).

[107] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali, Y. Yang, and Y. Zhou, *Deep learning scaling is predictable, empirically,* arXiv:1712.00409 (2017).

[108] O. Bousquet, S. Hanneke, S. Moran, R. van Handel, and A. Yehudayoff, *A theory of universal learning,* arXiv preprint arXiv:2011.04483 (2020).

[109] T. M. Cover, *Rates of convergence for nearest neighbor procedures,* in *Proceedings of the Hawaii International Conference on Systems Sciences*, Vol. 415 (1968).

[110] T. Cover and P. Hart, *Nearest neighbor pattern classification,* IEEE Trans. IT **13**, 21 (1967).

[111] D. Peterson, *Some convergence properties of a nearest neighbor decision rule,* IEEE Trans. IT **16**, 26 (1970).

[112] S.-i. Amari, *A universal theorem on learning curves,* Neural Networks **6**, 161 (1993).

[113] S. Amari, *Universal property of learning curves under entropy loss,* in *IJCNN*, Vol. 2 (1992) pp. 368–373 vol.2.

[114] M. Opper and D. Haussler, *Calculation of the learning curve of bayes optimal classification algorithm for learning a perceptron with noise,* in *COLT*, Vol. 91 (1991) pp. 75–87.

[115] S.-i. Amari, N. Fujita, and S. Shinomoto, *Four types of learning curves,* Neural Computation **4**, 605 (1992).

[116] S.-i. Amari and N. Murata, *Statistical Theory of Learning Curves under Entropic Loss Criterion,* Neural Computation **5**, 140 (1993).

[117] H. S. Seung, H. Sompolinsky, and N. Tishby, *Statistical mechanics of learning from examples,* Physical review A **45**, 6056 (1992).

[118] L. J. Savage, *The foundations of statistics* (John Wiley & Sons, Inc., 1954).

[119] I. J. Good, *On the principle of total evidence,* The British Journal for the Philosophy of Science **17**, 319 (1967).

[120] P. D. Grünwald and J. Y. Halpern, *When ignorance is bliss,* in *Proceedings of the 20th conference on Uncertainty in artificial intelligence* (2004) pp. 226–234.

[121] D. M. Tax and R. P. Duin, *Learning curves for the analysis of multiple instance classifiers,* in *S+SSPR* (Springer, 2008) pp. 724–733.

[122] X. L. Meng and X. Xie, *I Got More Data, My Model is More Refined, but My Estimator is Getting Worse! Am I Just Dumb?* Econometric Reviews **33**, 218 (2014).

[123] K. M. Ting, T. Washio, J. R. Wells, and S. Aryal, *Defying the gravity of learning curve: a characteristic of nearest neighbour anomaly detectors,* Machine Learning **106**, 55 (2017).

[124] G. M. Weiss and A. Battistin, *Generating well-behaved learning curves: An empirical study,* in *ICDATA* (2014).

[125] M. Loog, T. Viering, A. Mey, J. H. Krijthe, and D. M. Tax, *A brief prehistory of double descent,* PNAS **117**, 10625 (2020).

[126] W. L. Bryan and N. Harter, *Studies in the physiology and psychology of the telegraphic language,* Psychological Review **4**, 27 (1897).

[127] W. L. Bryan and N. Harter, *Studies on the telegraphic language: the acquisition of a hierarchy of habits,* Psychological Review **6**, 345 (1899).

[128] G. Vetter, M. Stadler, and J. D. Haynes, *Phase transitions in learning,* The Journal of Mind and Behavior , 335 (1997).

[129] S. Patarnello and P. Carnevali, *Learning networks of neurons with boolean logic,* Europhysics Letters **4**, 503 (1987).

[130] G. Györgyi, *First-order transition to perfect generalization in a neural network with binary synapses,* Physical Review A **41**, 7097 (1990).

[131] T. L. H. Watkin, A. Raut, and M. Biehl, *The Statistical Mechanics of Learning a Rule,* Reviews of Modern Physics **65**, 499 (1993).

[132] K. Kang, J.-H. Oh, C. Kwon, and Y. Park, *Generalization in a two-layer neural network,* Physical Review E **48**, 4805 (1993).

[133] M. Opper, *Statistical Mechanics of Learning : Generalization,* The Handbook of Brain Theory and Neural Networks , 20 (1995).

[134] H. Schwarze and J. A. Hertz, *Statistical Mechanics of Learning in a Large Committee Machine,* NeurIPS , 523 (1993).

[135] D. Hansel, G. Mato, and C. Meunier, *Memorization without generalization in a multilayered neural network,* Epl **20**, 471 (1992).

[136] M. Opper, *Learning to generalize,* Frontiers of Life **3**, 763 (2001).

[137] H. Seung, *Annealed theories of learning,* Neural Networks: The Statistical Mechanics Perspective, Proceedings of the CTP-PRSRI Joint Workshop on Theoretical Physics. Singapore, World Scientific (1995).

[138] H. Sompolinsky, *Theoretical issues in learning from examples,* in *NEC Research Symposium* (1993) pp. 217–237.

[139] M. Biehl and A. Mietzner, *Statistical mechanics of unsupervised learning,* Europhysics Letters **24**, 421 (1993).

[140] D. C. Hoyle and M. Rattray, *Statistical mechanics of learning multiple orthogonal signals: asymptotic theory and fluctuation effects,* Physical review E **75**, 016101 (2007).

[141] N. Ipsen and L. K. Hansen, *Phase transition in PCA with missing data: Reduced signal-to-noise ratio, not sample size!* in *ICML* (2019) pp. 2951–2960.

[142] R. A. Bhat, N. Jain, A. Vaidya, M. Palmer, T. Ahmed, D. M. Sharma, and J. Babani, *Adapting predicate frames for urdu propbanking,* in *Workshop on Language Technology for Closely Related Languages and Language Variants* (2014) pp. 47–55.

[143] P. Nakkiran, *More data can hurt for linear regression: Sample-wise double descent,* arXiv:1912.07242 (2019).

[144] M. Skurichina and R. P. Duin, *Stabilizing classifiers for very small sample sizes,* in *ICPR*, Vol. 2 (IEEE, 1996) pp. 891–896.

[145] J. H. Krijthe and M. Loog, *The peaking phenomenon in semi-supervised learning,* in *S+SSPR)* (Springer, 2016) pp. 299–309.

[146] V. Tresp, *The Equivalence between Row and Column Linear Regression*, Tech. Rep. (Siemens, 2002).

[147] P. Nakkiran, P. Venkat, S. Kakade, and T. Ma, *Optimal regularization can mitigate double descent,* arXiv:2003.01897 (2020).

[148] R. P. W. Duin, *Small sample size generalization,* 9th Scandinavian Conference on Image Analysis , 1 (1995).

[149] M. Skurichina and R. P. W. Duin, *Regularisation of Linear Classifiers by Adding Redundant Features,* Pattern Anal. Appl. **2**, 44 (1999).

[150] M. Opper and R. Urbanczik, *Universal learning curves of support vector machines,* Physical Review Letters **86**, 4410 (2001).

[151] S. Spigler, M. Geiger, S. dAscoli, L. Sagun, G. Biroli, and M. Wyart, *A jamming transition from under-to over-parametrization affects generalization in deep learning,* Journal of Physics A **52**, 474001 (2019).

[152] M. S. Advani and A. M. Saxe, *High-dimensional dynamics of generalization error in neural networks,* arXiv:1710.03667 (2017).

[153] T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani, *Surprises in high-dimensional ridgeless least squares interpolation,* arXiv:1903.08560 (2019).

[154] S. d'Ascoli, L. Sagun, and G. Biroli, *Triple descent and the two kinds of overfitting: Where & why do they appear?* arXiv:2006.03509 (2020).

[155] M. Loog and R. P. W. Duin, *The dipping phenomenon,* in *S+SSPR* (Hiroshima, Japan, 2012) pp. 310–317.

[156] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, *Large margin classifiers: convex loss, low noise, and convergence rates,* in *NeurIPS* (2004) pp. 1173–1180.

[157] S. Ben-david, D. Loker, N. Srebro, and K. Sridharan, *Minimizing the misclassification error rate using a surrogate convex loss,* ICML , 1863 (2012).

[158] J. Vanschoren, B. Pfahringer, and G. Holmes, *Learning from the past with experiment databases,* in *Pacific Rim International Conference on Artificial Intelligence* (Springer, 2008) pp. 485–496.

[159] G. Schohn and D. Cohn, *Less is more: Active learning with support vector machines,* in *ICML*, Vol. 2 (2000) p. 6.

[160] K. Konyushkova, R. Sznitman, and P. Fua, *Introducing geometry in active learning for image segmentation,* in *CVPR* (2015) pp. 2974–2982.

[161] M. Loog and Y. Yang, *An empirical investigation into the inconsistency of sequential active learning,* in *ICPR* (IEEE, 2016) pp. 210–215.

[162] Z. Wang, Z. Dai, B. Póczos, and J. Carbonell, *Characterizing and avoiding negative transfer,* in *CVPR* (2019) pp. 11293–11302.

[163] K. Weiss, T. M. Khoshgoftaar, and D. Wang, *A survey of transfer learning,* Journal of Big data **3**, 9 (2016).

[164] W. M. Kouw and M. Loog, *A review of domain adaptation without target labels,* TPAMI (2019).

[165] M. Loog, T. Viering, and A. Mey, *Minimizers of the empirical risk and risk monotonicity,* in *NeurISP* (2019) pp. 7478–7487.

[166] P. Grünwald and T. van Ommen, *Inconsistency of bayesian inference for misspecified linear models, and a proposal for repairing it,* Bayesian Analysis **12**, 1069 (2017).

[167] P. D. Grünwald and W. Kotłowski, *Bounds on individual risk for log-loss predictors,* JMLR **19**, 813 (2011).

[168] M. F. Al-Saleh and F. A. Masoud, *A note on the posterior expected loss as a measure of accuracy in bayesian methods,* Applied mathematics and computation **134**, 507 (2003).

[169] T. J. Viering, A. Mey, and M. Loog, *Making learners (more) monotone,* in *IDA* (Springer, 2020) pp. 535–547.

[170] Z. Mhammedi and H. Husain, *Risk-monotonicity in statistical learning,* arXiv preprint arXiv:2011.14126 (2020).

[171] R. A. Schiavo and D. J. Hand, *Ten more years of error rate research,* International Statistical Review **68**, 295 (2000).

[172] D. J. Hand, *Construction and assessment of classification rules* (Wiley, 1997).

[173] R. B. Anderson and R. D. Tweney, *Artifactual power curves in forgetting,* Memory & Cognition **25**, 724 (1997).

[174] A. Heathcote, S. Brown, and D. J. Mewhort, *The power law repealed: The case for an exponential law of practice,* Psychonomic bulletin & review **7**, 185 (2000).

[175] A. Mey, *A note on high-probability versus in-expectation guarantees of generalization bounds in machine learning,* arXiv:2010.02576 (2020).

[176] R. A. Fisher, *An absolute criterion for fitting frequency curves,* Messenger of Mathematics **41**, 155 (1912).

[177] S. M. Stigler, *The epic story of maximum likelihood,* Statistical Science **22**, 598 (2007).

5

MINIMIZERS OF THE EMPIRICAL RISK AND RISK MONOTONICITY

Plotting a learner's average performance against the number of training samples results in a learning curve. Studying such curves on one or more data sets is a way to get to a better understanding of the generalization properties of this learner. The behavior of learning curves is, however, not very well understood and can display (for most researchers) quite unexpected behavior. Our work introduces the formal notion of risk monotonicity*, which asks the risk to not deteriorate with increasing training set sizes in expectation over the training samples. We then present the surprising result that various standard learners, specifically those that minimize the empirical risk, can act* non*monotonically irrespective of the training sample size. We provide a theoretical underpinning for specific instantiations from classification, regression, and density estimation. Altogether, the proposed monotonicity notion opens up a whole new direction of research.*

A precursor to this work (see Appendix A) was presented as an open problem at COLT 2019 and has been published as an extended abstract in Volume 99 of the Proceedings of Machine Learning Research [1]. This work has been accepted at NeuRIPS 2019, Vancouver, Canada [2]. Appendix D contains the proofs of this work (page 187).

5.1. INTRODUCTION

Learning curves are an important diagnostic tool that provide researchers and practitioners with insight into a learner's generalization behavior [3]. Learning curves plot the (estimated) true performance against the number of training samples. Among other things, they can be used to compare different learners to each other. This can highlight the differences due to their complexity, with the simpler learners performing better in the small sample regime, while the more complex learners perform best with large sample sizes. In combination with a plot of their (averaged) resubstitution error (or training error), they can also be employed to diagnose underfitting and overfitting. Moreover, they can aid when it comes to making decision about collecting more data or not by extrapolating them to sample sizes beyond the ones available.

It seems intuitive that learners become better (or at least do not deteriorate) with more training data. With a bit more reservation, Shalev-Shwartz and Ben-David [3] state, for instance, that the learning curve "must start decreasing once the training set size is larger than the VC-dimension" (page 153). The large majority of researchers and practitioners (that we talked to) indeed take it for granted that learning curves show improved performance with more data. Any deviations from this they contribute to the way the experiments are set up, to the finite sample sizes one is dealing with, or to the limited number of cross-validation or bootstrap repetitions one carried out. It is expected that if one could sample a training set *ad libitum* and measure the learner's *true* performance over all data, such behavior disappears. That is, if one could indeed get to the performance in expectation over all test data and over all training samples of a particular size, performance supposedly improves with more data.

We formalize this behavior of expected improved performance in Section 5.3. As we will typically express a learner's efficiency in term of the expected loss, we will refer to this notation as *risk monotonicity*. Section 5.4 then continues with the main contribution of this work and demonstrates that various well-known empirical risk minimizers can display nonmonotonic behavior. Moreover, we show that for these learners this behavior can persist indefinitely, *i.e.*, it can occur at any sample size. *Note*: Appendix D contains the proofs of this work (page 187). Section 5.5 provides some experimental evidence for some cases of interest that have, up to now, resisted any deeper theoretical analysis. Section 5.6 then provides a discussion and concludes the work. In this last section, among others, we contrast our notion of risk monotonicity to that of PAC-learnability, note that these are two essentially different concepts, and consider various research questions of interest to further refine our understanding of learning curves. Though many will probably find our findings surprising, counterintuitive behavior of the learning curve has been reported before in various other settings. Section 5.2 goes through these and other related works and puts our contribution in perspective.

5.2. EARLIER WORK AND ITS RELATION TO THE CURRENT

We split up our overview into the more regular works that characterize monotonic behavior and those that identify the existence of nonmonotonic behavior.

5.2.1. THE MONOTONIC CHARACTER OF LEARNING CURVES

Many of the studies into the behavior of learning curves stem from the end of the 1980s and the beginning of the 1990s and were carried out by Tishby, Haussler, and others [4–9]. These early investigations were done in the context of neural networks and in their analyses typically make use of tools from statistical mechanics. A statistical inference approach is studied by Amari et al. [10] and Amari and Murata [11], who demonstrate the typical power-law behavior of the asymptotic learning curve. Haussler et al. [12] bring together many of the techniques and results from the aforementioned works. At the same time, they advance the theory for learning curves and provide an overview of the rather diverse, though still monotonic, behavior they can exhibit. In particular, the curve may display multiple steep and sudden drops in the risk.

Already in 1979, Micchelli and Wahba [13] provide a lower bound for learning curves of Gaussian processes. Only at the end of the 1990s and beginning of the 2000s, the overall attention shifted from neural networks to Gaussian processes. In this period, various works were published that introduce approximations and bounds [14–18]. Different types of techniques were employed in these analyses, some of which again from statistical mechanics. The main caveat, when it comes to the results obtained, is the assumption that the model is correctly specified.

The focus of [19] is on support vector machines. They develop efficient procedures for an extrapolation of the learning curve, so that if only limited computational resources are available, these can possibly be assigned to the most promising approaches. It is assumed that, for large enough training set sizes, the error rate converges towards a stable value following a power-law. This behavior was established to hold in many of the aforementioned works. The ideas that [19] put forward have found use in specific applications (see, for instance, [20]) and can count on renewed interest these days, especially in combination with flop gobbling neural networks (see, for instance, [21]).

All of the aforementioned works study and derive learning curve behavior that shows no deterioration with growing training set sizes, even though they may be described as "learning curves with rather curious and dramatic behavior" [12]. Our work identifies aspects that are more curious and more dramatic: with a larger training set, performance can deteriorate, even in expectation.

5.2.2. EARLY NOTED NONMONOTONIC BEHAVIOR

Opper [22] already made the case that nonmonotonic behavior could be expected by studying the theoretical limit behavior of learning curves for very large single layer neural networks (see also [23] and [24]). The first to demonstrate this behavior empirically was probably Duin [25], who looked at the error rate of the model referred to as Fisher's linear discriminant. In this context, Fisher's linear discriminant is used as a classifier and equivalent to the two-class linear classifier that is obtained by optimizing the squared loss. This can be solved by regressing the input feature vectors onto a $-1/+1$ encoding of the class labels. In case the number of training samples is smaller than or equal to the number of input dimensions, one needs to deal with the inverse of singular matrices and typically resorts to the use of the Moore-Penrose pseudo-inverse. In this way, the minimum norm solution is obtained [26]. It is exactly in this underdetermined setting, as the number of

training samples approaches the dimensionality, that the error rate will be increasing. Other examples of exactly this type of nonmonotonic behavior have been reported. Worth mentioning are classifiers built based on the lasso [27] and two recent works that have triggered renewed attention to this subject in the neural networks community [28, 29]. The classifier reaching a maximum error rate when the sample size transits from an underspecified to an overspecified setting is originally referred to as peaking (see also [30]). The two recent works above rename it and use the terms double descent and jamming.

A completely different phenomenon, and yet other way in which learning curves can be nonmonotonic, is described by Loog and Duin [31]. They show that there are learning problems for which specific classifiers attain their optimal expected 0-1 loss at a finite sample size. That is, on such problems, these classifiers perform essentially worse with an infinite amount of training data compared to some finite training set sizes. The behavior is referred to as dipping, following the shape of the error rate's learning curve. In the context of (safe) semi-supervised learning, Loog [32] then argues that if one cannot even guarantee improvements in 0-1 loss when receiving more labeled data, this is certainly impossible with unlabeled data. When evaluating in terms of the loss the model optimizes, however, one can get to demonstrable improvements and essentially solve the safe semi-supervised leaning problem [32–34]. Our work shows, however, that also when one looks at the loss the learner optimizes, there may be no performance guarantees.

The dipping behavior hinges both on the fact that the model is misspecified (i.e., the Bayes-optimal estimate is not in the class of models considered) and that the classifier does not optimize what it is ultimately evaluated with. That this setting can cause problems, e.g., convergence to the wrong solution, had already been demonstrated for maximum likelihood by Devroye et al. [35]. If the model class is flexible enough, this discrepancy disappears in many a setting. This happens, for instance, for the class of classification-calibrated surrogate losses [36]. Note, however, that Devroye et al. [35] conjecture that consistent rules that are expected to perform better with increasing training sizes (referred to as smart rules) do not exist. Ben-David et al. [37] analyze the consequence of the mismatch between surrogate and zero-one loss in some more detail and provide another example of a problem distribution on which such classifiers would dip.

Our results strengthen or extend the above findings in the following ways. First of all, we show that nonmonotonic behavior can occur in the setting where the complexity of the learner is small compared to the training set size. Therefore, the reported behavior is not due to jamming or peaking. Secondly, we are going to evaluate our learners by means of the loss they actually optimize for. If we look at the linear classifier that optimizes the hinge loss, for instance, we will study its learning curve for the hinge loss as well. In other words, there is no discrepancy between the objective used during training and the loss used at test time. Therefore, possibly odd behavior cannot be explained by dipping. As a third, we do not only look at classification and regression but also consider density estimation and (negative) log-likelihood estimation in particular.

5.3. RISK MONOTONICITY

We come to a formal definition of the intuition that with one additional instance a learner should improve its performance in expectation over the training set. The next sections then

study various learners with the notions developed here. First, however, some notations and prior definitions are provided.

5.3.1. PRELIMINARIES

We let $S_n = (z_1, \ldots, z_n)$ be a training set of size n, sampled i.i.d. from a distribution P over a general domain $\mathcal{Z}$. Also given is a hypothesis class H and a loss function $\ell : \mathcal{Z} \times H \to \mathbb{R}$ through which the performance of a hypothesis $h \in H$ is measured. The objective is to minimize the expected loss or risk under the distribution P, which is given by

$$R_P(h) := \mathop{\mathbb{E}}_{z \sim P} \ell(z, h). \tag{5.1}$$

A learner A is a particular mapping from the set of all samples $\mathcal{S} := \mathcal{Z} \cup \mathcal{Z}^2 \cup \mathcal{Z}^3 \cup \ldots$ to elements from the prespecified hypothesis class H. That is, $A : \mathcal{S} \to H$. We are particularly interested in learners A_{erm} that provide a solution which minimizes the empirical risk $\hat{R}_{S_n}$ over the training set:

$$A_{\text{erm}}(S_n) := \operatorname*{argmin}_{h \in H} \hat{R}_{S_n}(h), \tag{5.2}$$

with

$$\hat{R}_{S_n}(h) := \frac{1}{n} \sum_{i=1}^{n} \ell(z_i, h). \tag{5.3}$$

Most common classification, regression, and density estimation problems can be formulated in such terms. Examples are the earlier mentioned Fisher's linear discriminant, support vector machines, and Gaussian processes, but also maximum likelihood estimation, linear regression, and the lasso can be cast in similar terms.

5.3.2. DEGREES OF MONOTONICITY

The basic definition is the following.

Definition 1 (local monotonicity)**.** *A learner A is (P, ℓ, n)-monotonic with respect to a distribution P, a loss ℓ, and an integer $n \in \mathbb{N} := \{1, 2, \ldots\}$ if*

$$\mathop{\mathbb{E}}_{S_{n+1} \sim D^{n+1}} [R_P(A(S_{n+1})) - R_P(A(S_n))] \leq 0. \tag{5.4}$$

This expresses exactly how we would expect a learner to behave locally (i.e., at a specific training sample size n): given one additional training instance, we expect the learner to improve. Based on our definition of local monotonicity, we can construct stronger desiderata that may be of more interest.

The two entities we would like to get rid of in the above definition are n and P. The former, because we would like our learner to act monotonically irrespective of the sample size. The latter, because we typically do not know the underlying distribution. For now, getting rid of the loss ℓ is maybe too much to ask for. First of all, not all losses are compatible with one another, as they may act on different types of $z \in \mathcal{Z}$ and $h \in H$. But even if they take the same types of input, a learner is typically designed to minimize one specific loss and there seems to be no direct reason for it to be monotonic in terms of another. It

seems less likely, for example, that an SVM is risk monotonic in terms of the squared loss. (We will nevertheless briefly return to this matter in Section 5.6.) We exactly focus on the empirical risk minimizers as they seem to be the most appropriate candidates to behave monotonically in terms of their own loss.

Though we typically do not know P, we do know in which domain $\mathcal{Z}$ we are operating. Therefore, the following definition is suitable.

Definition 2 (local $\mathcal{Z}$-monotonicity)**.** *A learner A is (locally) $(\mathcal{Z}, \ell, n)$-monotonic with respect to a loss ℓ and an integer $n \in \mathbb{N}$ if, for all distributions P on $\mathcal{Z}$, it is (P, ℓ, n)-monotonic.*

When it comes to n, the peaking phenomenon shows that, for some learners, it may be hopeless to demand local monotonicity for all $n \in \mathbb{N}$. What we still can hope to find is an $N \in \mathbb{N}$, such that for all $n \geq N$, we find the learner to be locally risk monotonic. As properties like peaking may change with the dimensionality—the complexity of the classifier is generally dependent on it, the choice for N will typically have to depend on the domain.

Definition 3 (weak $\mathcal{Z}$-monotonicity)**.** *A learner A is weakly $(\mathcal{Z}, \ell, N)$-monotonic with respect to a loss ℓ if there is an integer $N \in \mathbb{N}$ such that for all $n \geq N$, the learner is locally $(\mathcal{Z}, \ell, n)$-monotonic.*

Given the domain, one may of course be interested in the smallest N for which weak $\mathcal{Z}$-monotonicity is achieved. If it does turn out that N can be set to 1, the learner is said to be globally $\mathcal{Z}$-monotonic.

Definition 4 (global $\mathcal{Z}$-monotonicity)**.** *A learner A is globally $(\mathcal{Z}, \ell)$-monotonic with respect to a loss ℓ if for every integer $n \in \mathbb{N}$, the learner is locally $(\mathcal{Z}, \ell, n)$-monotonic.*

5.4. THEORETICAL RESULTS

We consider the hinge loss, the squared loss, and the absolute loss and linear models that optimize the corresponding empirical loss. In essence, we demonstrate that, there are various domains $\mathcal{Z}$ for which for any choice of N, these learners are *not* weakly $(\mathcal{Z}, \ell, N)$-monotonic. For the log-likelihood, we basically prove the same: there are standard learners for which the (negative) log-likelihood is not weakly $(\mathcal{Z}, \ell, N)$-monotonic for any N. The first three losses can all be used to build classifiers: the first is at the basis of SVMs, while the second gives rise to Fisher's linear discriminant in combination with linear hypothesis classes. The second and third loss are of course also employed in regression. The log-likelihood is standard in density estimation.

5.4.1. LEARNERS THAT DO BEHAVE MONOTONICALLY

Before we actually move to our negative results, we first provide examples that point in a positive direction. The first learner is provably risk monotonic over a large collection of domains. The second learner, the memorize algorithm, is a monotonic learner taken from [38].

Fitting a normal distribution with fixed covariance and unknown mean. Let Σ be a $d \times d$ invertible covariance matrix,

$$H := \left\{ z \mapsto \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp(-\tfrac{1}{2}(z-\mu)^T \Sigma^{-1}(z-\mu)) \,\middle|\, \mu \in \mathbb{R}^d \right\}, \tag{5.5}$$

$\mathcal{Z} \subset \mathbb{R}^d$, and take the loss to equal the negative log-likelihood.

Theorem 8. *If $\mathcal{Z}$ is bounded, the learner A_{erm} is globally $(\mathcal{Z}, \ell)$-monotonic.*

Remark 1. *Using similar arguments, one can show that the learner with $H = \mathbb{R}^d$ and Mahalanobis loss $\ell(z,h) = ||z-h||_\Sigma^2 := (z-h)^T\Sigma(z-h)$, with Σ a positive semi-definite matrix, is globally $(\mathcal{Z}, \ell)$-monotonic as well as long as $\mathcal{Z}$ is bounded.*

The memorize algorithm [38]. When evaluated on a test input object that is also present in the training set, this classifier returns the label of said training object. In case multiple training examples share the same input, the majority voted label is returned. In case the test object is not present in the training set, a default label is returned. This learner is monotonic for any distribution under the zero-one loss. Similarly, any histogram rule with fixed partitions is monotone, which follows from properties of the binomial distribution [35].

5.4.2. LEARNERS THAT DO NOT BEHAVE

To show for various learners that they do not always behave risk monotonically, we construct specific discrete distributions for which we can explicitly proof nonmonotonicity. What leads to the sought-after counterexamples in our case, is a distribution where a small fraction of the density is located relatively far away from the origin. In particular, shrinking the probability of this fraction towards 0 leads us to the lemma below. It is used in the subsequent proofs, but is also of some interest in itself.

Lemma 2. *Let $\mathcal{Z} := \{a, b\}$ be a domain with two elements from $\mathbb{R}$, let*

$$S_{n-k}^k := (\underbrace{a, \ldots, a}_{k \text{ elements}}, \underbrace{b, \ldots, b}_{n-k \text{ elements}}) \tag{5.6}$$

be a training set with n samples, and let $h_{n-k}^k := A_{\text{erm}}(S_{n-k}^k)$. If

$$-\ell(b, h_{n+1}^0) + (n+1)\ell(b, h_n^1) - n\ell(b, h_{n-1}^1) > 0, \tag{5.7}$$

then A_{erm} is not locally $(\mathcal{Z}, \ell, n)$-monotonic.

Remark 2. *For many losses, we have, in fact, that $\ell(b, h_n^0) = \ell(b, h_{n+1}^0) = 0$, which further simplifies the difference of interest to $(n+1)\ell(b, h_n^1) - n\ell(b, h_{n-1}^1)$.*

In a way, the above lemma and remark show that if the learning of the single point b does not happen fast enough, local monotonicity cannot be guaranteed. Section 5.6 will briefly return to this point.

Linear hypotheses, squared loss, absolute loss, and hinge loss. We consider linear models without bias in d dimensions, so take $\mathcal{Z} = \mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^d \times \mathbb{R}$ and $H = \mathbb{R}^d$. Though not crucial to our argument, we select the minimum-norm solution in the underdetermined case. For the squared loss, we have $\ell(z, h) = (x^T h - y)^2$ for any $z = (x, y) \in \mathcal{Z}$. The absolute loss is given by $\ell(z, h) = |x^T h - y|$ and the hinge loss is defined as $\ell(z, h) = \max(0, 1 - yx^T h)$. Both the absolute loss and the squared loss can be used for regression and classification. The hinge loss is appropriate only for the classification setting. Therefore, though the rest of the setup remains the same, outputs are limited to the set $\mathcal{Y} = \{-1, +1\}$ for the hinge loss.

Theorem 9. *Consider a linear A_{erm} without intercept and assume it either optimizes the squared, the absolute, or the hinge loss. Assume $\mathcal{Y}$ contains at least one nonzero element. If there exists an open ball B_0 that contains the origin, such that $B_0 \subset \mathcal{X}$, then this risk minimizer is* not *weakly $(\mathcal{Z}, \ell, N)$-monotonic for any $N \in \mathbb{N}$.*

Fitting a normal distribution with fixed mean and unknown variance (in one dimension). We follow up on the example where we fitted a normal distribution with fixed covariance and unknown mean. We limit ourselves, however, to one dimension only and, more importantly, now take the variance to be the unknown, while fixing the mean (to 0, arbitrarily). Specifically, let $H := \{z \mapsto \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2} z^2) | \sigma > 0\}$, $\mathcal{Z} \subset \mathbb{R}$, and take the loss to equal the negative log-likelihood.

Theorem 10. *If there exists an open ball B_0 that contains the origin, such that $B_0 \subset \mathcal{Z}$, then estimating the variance of a one-dimensional normal density is* not *weakly $(\mathcal{Z}, \ell, N)$-monotonic for any $N \in \mathbb{N}$.*

5.5. EXPERIMENTAL EVIDENCE

Our results from the previous section, already show cogently that the behavior of the learning curve can be interesting to study. Here we complement our theoretical findings with a few illustrative experiments[1] to strengthen this point even further. The results can be found in Figure 5.1, which displays (numerically) exact learning curves for a couple of different settings. The computation is exact, because in the risk computation we enumerate all possible training sets, and therefore we do not display standard errors of the curves.

The input space considered for all our examples is one-dimensional. The experiment in Subfigure 5.1b relies on the absolute loss, while all other make use of the squared loss. In addition, Subfigures 5.1a, 5.1b, and 5.1c consider distributions with two points: $a = (1, 1)$ and $b = (\frac{1}{10}, 1)$ with the first coordinate the input and the second the corresponding output. Different plots use different values for the probability of observing a. For Subfigure 5.1a, $P(a) = 0.00001$, Subfigure 5.1b uses $P(a) = 0.1$, and Subfigure 5.1c takes $P(a) = 0.01$. For Subfigure 5.1c, we also studied the effect of a small amount of standard L_2-regularization decreasing with training size ($\mu = \frac{0.01}{n}$), leading to the regularized solution A_{reg}. The distribution for Subfigure 5.1d is slightly different and supported on three points: $a = (1, 1)$, $b = (\frac{1}{10}, -1)$, and $c = (-1, 1)$, with again the first coordinate as the input and the second the corresponding output. In this case, $P(a) = 0.01$, $P(b) = 0.01$, and $P(c) = 0.98$.

[1] Code is available through `https://github.com/tomviering/RiskMonotonicity`

This last experiment concerns least squares regression with a bias term: a setting we have not been able to analyze theoretically up to this point.

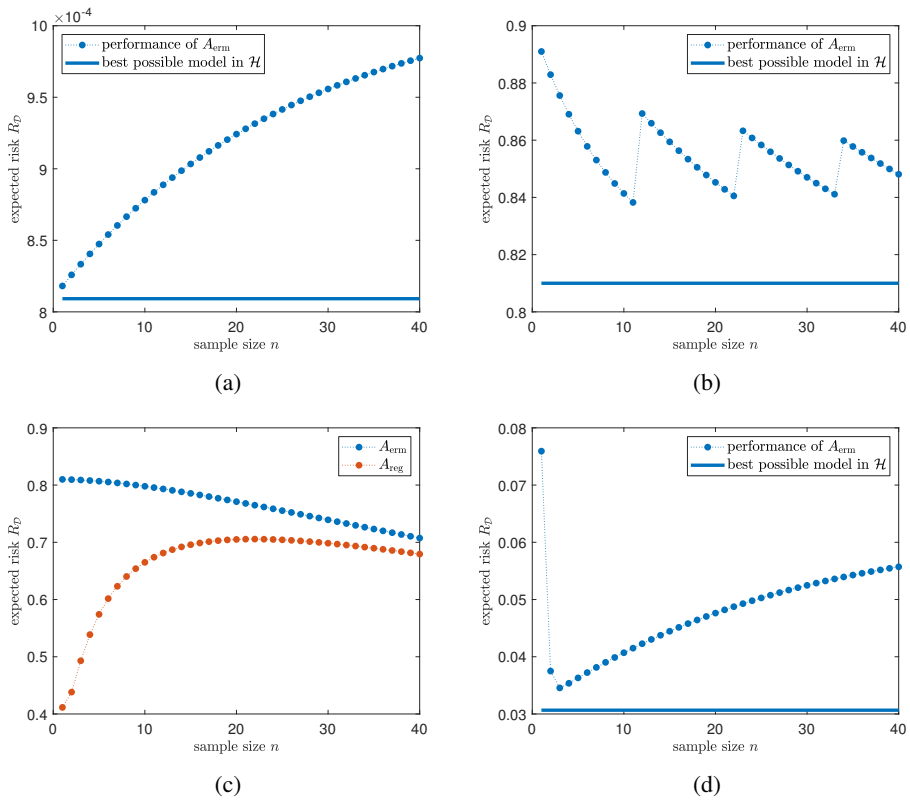


Figure 5.1.: Learning curves (average risk against training set size) for some one-dimensional problems. Subfigure (a) is based on squared loss, no intercept; (b) on absolute loss, no intercept; (c) on squared loss, no intercept (with and without regularization); (d) on squared loss with intercept. The solid line, indicates the risk the learner attains in the limit of an infinite training set size.

Most salient is probably the serrated and completely nonmonotonic behavior of the learning curve for the absolute loss in Figure 5.1b. Of interest as well is that regularization does not necessarily solve the problem. Subfigure 5.1c even shows it can make it worse: A_{reg} gives nonmonotonic behavior, while A_{erm} is monotonic under the same distribution (cf. [39]). Subfigure 5.1a illustrates clearly how dramatic the expected squared loss can grow with more data.

In the final example in Figure 5.1d, as already noted, we consider linear regression with the squared loss that includes a bias term in combination with the distribution supported on three points. This example is of interest because the usual configuration for standard learners includes such bias term and one could get the impression from our theoretical res-

ults (and maybe in particular the proofs) that the origin plays a major role in the bad behavior of some of the learners. But as can be observed here, adding an intercept, and therefore taking away the possibly special status of the origin does not make risk nonmonotonicity go away.

5.6. DISCUSSION AND CONCLUSION

It should be clear that this paper does not get to the bottom of the learning-curve issue. In fact, one of the reasons of this work is to bring it to the attention of the community. We are convinced that it raises a lot of interesting and interrelated problems that may go far beyond the initial analyses we offer here. Further study should bring us to a better understanding of how learning curves can actually act, which, in turn, should enable practitioners to better interpret and anticipate them.

What this work does convey is that learning curves can (provably) show some rather counterintuitive and surprising behavior. In particular, we have demonstrated that least squares regression, regression with the absolute loss, linear models trained with the hinge loss, and likelihood estimation of the variance of a normal distribution can all suffer from nonmonotonic behavior, even when evaluated with the loss they optimize for. All of these are standard learners, using standard loss functions.

Anyone familiar with the theory of PAC learning may wonder how our results can be reconciliated with the bounds that come from this theory. At a first glance, our observations may seem to contradict this theory. Learning theory dictates that if the hypothesis class has finite VC-dimension, the excess risk ϵ of ERM will drop as $\epsilon = O(\frac{1}{n})$ in the realizable case and as $\epsilon = O(\frac{1}{\sqrt{n}})$ in the agnostic case [3, 40]. Thus PAC bounds give an upper bound on the excess risk ϵ that will be tighter given more samples. PAC bounds hold with a particular probability, but we are concerned with the risk in expectation. Even bounds that hold in expectation over the training sample will, however, not rule out nonmonotonic behavior. This is because in the end the guarantees from PAC learning are indeed merely bounds. Our analysis show that within those bounds, we cannot always expect risk monotonic behavior. In fact, learning problems of all four possible combinations exist: not PAC-learnable and monotonic, PAC-learnable and not monotonic, etc. For instance, the memorize algorithm (end of Subsection 5.4.1) is monotone, while it has infinite VC-dimension and so is not PAC-learnable.

In light of the learning rates mentioned above, we wonder whether there are deeper links with Lemma 2 (see also Remark 2). Rewrite Equation (5.7) to find that we do not have local monotonicity at n in case

$$\frac{-\frac{\ell(b,h^0_{n+1})}{n+1} + \ell(b,h^1_n)}{\ell(b,h^1_{n-1})} > \frac{n}{n+1}. \tag{5.8}$$

With n large enough, we can ignore the first term in the numerator. So if a learner, in this particular setting, does not learn an instance b at least at a rate of $\frac{n}{n+1}$ in terms of the loss, it will display nonmonotonic behavior. According to learning theory, for agnostic learners, the fraction between two subsequent losses is of the order $\sqrt{\frac{n}{n+1}}$, which is always larger than $\frac{n}{n+1}$ for $n > 0$. Can one therefore generally expect nonmonotonic behavior for any

agnostic learner? Our normal mean estimation problem shows it cannot. But then, what is the link, if any?

As already hinted at in the introduction, our findings may also warrant revisiting the results obtained in [32–34]. These works show that there are some semi-supervised learners that allow for essentially improved performance over the supervised learner, i.e., these are truly safe. Though this is the transductive setting, this may in a sense just show how strong these results are. In the end, their estimation procedures are really rather different from empirical risk minimization, but it does beg the question whether similar constructs can be used to get to risk monotonic procedures in the supervised case.

Another question, related to the last remark above, seems of interest: could it be that the use of particular losses at training time leads to monotonic behavior at test time? Or can regularization still lead to more monotonic behavior, e.g., by explicitly limiting H? Maybe particular (upper-bounding) convex losses could turn out to behave risk monotonic in terms of specific nonconvex losses? Dipping seems to show, however, that this may very well not be the case. Results concerning smart rules, i.e., classifiers that act monotonically in terms of the error rate [35], seem to point in the same direction. So should we expect it to be the other way round? Can nonconvex losses bring us monotonicity guarantees for convex ones? Of course, monotonicity properties of *nonconvex* learners are also of interest to study in their own respect.

An ultimate goal would of course be to fully characterize when one can have risk monotonic behavior and when not. At this point we do not have a clear idea to what extent this would at all be possible. We were, for instance, not able to analyze some standard, seemingly simple cases, e.g., simultaneously estimating the mean and the variance of a normal model. And maybe we can only get to rather weak results. Only knowledge about the domain may turn out to be insufficient and we need to make assumptions on the class of distributions we are dealing with (leading to some notion of weakly P-monotonicity?). For a start, we could study likelihood estimation under correctly specified models, for which generally there turn out to be remarkably few finite-sample results. One can also wonder whether it is possible to find salient distributional properties that can be specifically related to the overall shape of the learning curve (see, for instance, [12]).

All in all, we believe that our theoretical results, strengthened by some illustrative examples, show that the monotonicity of learning curves is an interesting and nontrivial property to study.

5.7. BIBLIOGRAPHY

[1] T. Viering, A. Mey, and M. Loog, *Open problem: Monotonicity of learning,* in *Proceedings of the Thirty-Second Conference on Learning Theory*, Proceedings of Machine Learning Research, Vol. 99, edited by A. Beygelzimer and D. Hsu (Phoenix, USA, 2019) pp. 3198–3201.

[2] M. Loog, T. Viering, and A. Mey, *Minimizers of the empirical risk and risk monotonicity,* Advances in Neural Information Processing Systems **32** (2019).

[3] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms* (Cambridge University Press, 2014).

[4] N. Tishby, E. Levin, and S. A. Solla, *Consistent inference of probabilities in layered networks: Predictions and generalization,* in *International Joint Conference on Neural Networks*, Vol. 2 (1989) pp. 403–409.

[5] E. Levin, N. Tishby, and S. A. Solla, *A statistical approach to learning and generalization in layered neural networks,* Proceedings of the IEEE **78**, 1568 (1990).

[6] H. Sompolinsky, N. Tishby, and H. S. Seung, *Learning from examples in large neural networks,* Physical Review Letters **65**, 1683 (1990).

[7] M. Opper and D. Haussler, *Calculation of the learning curve of bayes optimal classification algorithm for learning a perceptron with noise,* in *Proceedings of the fourth annual workshop on Computational learning theory* (Morgan Kaufmann Publishers Inc., 1991) pp. 75–87.

[8] H. Seung, H. Sompolinsky, and N. Tishby, *Statistical mechanics of learning from examples,* Physical Review A **45**, 6056 (1992).

[9] D. Haussler, M. Kearns, M. Opper, and R. Schapire, *Estimating average-case learning curves using bayesian, statistical physics and vc dimension methods,* in *Advances in Neural Information Processing Systems* (1992) pp. 855–862.

[10] S.-i. Amari, N. Fujita, and S. Shinomoto, *Four types of learning curves,* Neural Computation **4**, 605 (1992).

[11] S.-I. Amari and N. Murata, *Statistical theory of learning curves under entropic loss criterion,* Neural Computation **5**, 140 (1993).

[12] D. Haussler, M. Kearns, H. S. Seung, and N. Tishby, *Rigorous learning curve bounds from statistical mechanics,* Machine Learning **25**, 195 (1996).

[13] C. A. Micchelli and G. Wahba, *Design Problems for Optimal Surface Interpolation*, Tech. Rep. 565 (Department of Statistics, Wisconsin University, 1979).

[14] M. Opper, *Regression with Gaussian processes: Average case performance,* in *Theoretical aspects of neural computation: A multidisciplinary perspective* (Springer, 1998) pp. 17–23.

[15] P. Sollich, *Learning curves for Gaussian processes,* in *Advances in Neural Information Processing Systems* (1999) pp. 344–350.

[16] M. Opper and F. Vivarelli, *General bounds on bayes errors for regression with Gaussian processes,* in *Advances in Neural Information Processing Systems* (1999) pp. 302–308.

[17] C. K. Williams and F. Vivarelli, *Upper and lower bounds on the learning curve for Gaussian processes,* Machine Learning **40**, 77 (2000).

[18] P. Sollich and A. Halees, *Learning curves for Gaussian process regression: Approximations and bounds,* Neural Computation **14**, 1393 (2002).

[19] C. Cortes, L. D. Jackel, S. A. Solla, V. N. Vapnik, and J. S. Denker, *Learning curves: Asymptotic values and rate of convergence,* in *Advances in Neural Information Processing Systems* (1994) pp. 327–334.

[20] P. Kolachina, N. Cancedda, M. Dymetman, and S. Venkatapathy, *Prediction of learning curves in machine translation,* in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1* (Association for Computational Linguistics, 2012) pp. 22–30.

[21] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali, Y. Yang, and Y. Zhou, *Deep learning scaling is predictable, empirically,* arXiv preprint arXiv:1712.00409 (2017).

[22] M. Opper, *Statistical mechanics of learning: Generalization,* in *The Handbook of Brain Theory and Neural Networks* (1995) pp. 922–925.

[23] M. Opper and W. Kinzel, *Statistical mechanics of generalization,* in *Models of Neural Networks III* (Springer, 1996) pp. 151–209.

[24] M. Opper, *Learning to generalize,* Frontiers of Life **3**, 763 (2001).

[25] R. P. Duin, *Small sample size generalization,* in *Proceedings of the Scandinavian Conference on Image Analysis*, Vol. 2 (1995) pp. 957–964.

[26] A. J. Smola, P. J. Bartlett, D. Schuurmans, and B. Schölkopf, *Advances in Large Margin Classifiers* (MIT Press, 2000).

[27] N. Krämer, *On the peaking phenomenon of the lasso in model selection,* arXiv preprint arXiv:0904.4416 (2009).

[28] M. Belkin, D. Hsu, S. Ma, and S. Mandal, *Reconciling modern machine learning and the bias-variance trade-off,* arXiv preprint arXiv:1812.11118 (2018).

[29] S. Spigler, M. Geiger, S. d'Ascoli, L. Sagun, G. Biroli, and M. Wyart, *A jamming transition from under-to over-parametrization affects loss landscape and generalization,* arXiv preprint arXiv:1810.09665 (2018).

[30] R. P. Duin, *Classifiers in almost empty spaces,* in *Proceedings of the 15th International Conference on Pattern Recognition*, Vol. 2 (IEEE, 2000) pp. 1–7.

[31] M. Loog and R. P. Duin, *The dipping phenomenon,* in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)* (Springer, 2012) pp. 310–317.

[32] M. Loog, *Contrastive pessimistic likelihood estimation for semi-supervised classification,* IEEE Transactions on Pattern Analysis and Machine Intelligence **38**, 462 (2016).

[33] J. H. Krijthe and M. Loog, *Projected estimators for robust semi-supervised classification,* Machine Learning **106**, 993 (2017).

[34] J. H. Krijthe and M. Loog, *The pessimistic limits and possibilities of margin-based losses in semi-supervised learning,* in *Advances in Neural Information Processing Systems* (2018) pp. 1790–1799.

[35] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition* (Springer, 1996).

[36] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, *Convexity, classification, and risk bounds,* Journal of the American Statistical Association **101**, 138 (2006).

[37] S. Ben-David, D. Loker, N. Srebro, and K. Sridharan, *Minimizing the misclassification error rate using a surrogate convex loss,* in *Proceedings of the 29th International Conference on Machine Learning* (2012) pp. 83–90.

[38] S. Ben-David, N. Srebro, and R. Urner, *Universal learning vs. no free lunch results,* in *Philosophy and Machine Learning Workshop NIPS* (2011).

[39] P. D. Grünwald and W. Kotłowski, *Bounds on individual risk for log-loss predictors,* in *Proceedings of the 24th Annual Conference on Learning Theory* (2011) pp. 813–816.

[40] V. N. Vapnik, *Statistical Learning Theory* (Wiley, 1998).

6

MAKING LEARNERS (MORE) MONOTONE

Learning performance can show non-monotonic behavior. That is, more data does not necessarily lead to better models, even on average. We propose three algorithms that take a supervised learning model and make it perform more monotone. We prove consistency and monotonicity with high probability, and evaluate the algorithms on scenarios where non-monotone behaviour occurs. Our proposed algorithm MT_{HT} makes less than 1% *non-monotone decisions on MNIST while staying competitive in terms of error rate compared to several baselines. Our code is available at* `https://github.com/tomviering/monotone`.

This work has been accepted at IDA 2020, Virtual Conference [1].

6.1. INTRODUCTION

It is a widely held belief that more training data usually results in better generalizing machine learning models—cf. [2, 3] for instance. Several learning problems have illustrated, however, that more training data can lead to worse generalization performance [4–6]. For the peaking phenomenon [4], this occurs exactly at the transition from the underparametrized to the overparametrized regime. This double-descent behavior has found regained interest in the context of deep neural networks [7, 8], since these models are typically overparametrized. Recently, also several new examples have been found, where in quite simple settings more data results in worse generalization performance [9, 10].

It can be difficult to explain to a user that machine learning models can actually perform worse when more, possibly expensive to collect data has been used for training. Besides, it seems generally desirable to have algorithms that guarantee increased performance with more data. How to get such a guarantee? That is the question we investigate in this work and for which we use learning curves. Such curves plot the expected performance of a learning algorithm versus the amount of training data.[1] In other words, we wonder how we can make learning curves monotonic: meaning that the expected performance deteriorates less often or by smaller amounts, ideally always improving.

The core approach to make learners monotone is that, when more data is gathered and a new model is trained, this newly trained model is compared to the currently adopted model that was trained on less data. Only if the new model performs better should it be used. We introduce several wrapper algorithms for supervised classification techniques that use the holdout set or cross-validation to make this comparison. Our proposed algorithm MT_{HT} uses a hypothesis test to switch if the new model improves significantly upon the old model. Using guarantees from the hypothesis test we can prove that the resulting learning curve is monotone with high probability. We empirically study the effect of the parameters of the algorithms and benchmark them on several datasets including MNIST [11] to check to what degree the learning curves become monotone.

This work is organized as follows. The notion of monotonicity of learning curves is reviewed in Section 6.2. We introduce our approaches and algorithms in Section 6.3, and prove consistency and monotonicity with high probability in Section 6.4. Section 6.5 provides the empirical evaluation. We discuss the main findings of our results in Section 6.6 and end with the most important conclusions.

6.2. SETTING AND THE DEFINITION OF MONOTONICITY

We consider the setting where we have a learner that now and then receives data and that is evaluated over time. The question is then, how to make sure that the performance of this learner over time is monotone—or in other words, how can we guarantee that this learner over time improves its performance?

We analyze this question in a (frequentist) classification framework. We assume there exists an (unknown) distribution P over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the input space (features) and $\mathcal{Y}$ is the output space (classification labels). To simplify the setup we operate in rounds indicated by i, where $i \in \{1, \ldots, n\}$. In each round, we receive a batch of samples S^i that is sampled i.i.d. from P. The learner L can use this data in combination with data from

[1] Not to be confused with training curves, where the loss versus epochs (optimization iterations) is plotted.

previous rounds to come up with a hypothesis h_i in round i. The hypothesis comes from a hypothesis space H. H is not to be confused with a hypothesis H_0 or H_1, which will be defined later for our hypothesis test. We consider learners L that, as subroutine, use a supervised learner $A : \mathscr{S} \to H$, where $\mathscr{S}$ is the space of all possible training sets.

We measure performance by the error rate. The true error rate on P equals

$$\epsilon(h_i) = \int_{x \in \mathscr{X}} \sum_{y \in \mathscr{Y}} l_{01}(h_i(x), y) dP(x, y) \tag{6.1}$$

where l_{01} is the zero-one loss. We indicate the empirical error rate of h on a sample S as $\hat{\epsilon}(h, S)$. We call n rounds a run. The true error of the returned h_i by the learner L in round i is indicated by ϵ_i, all the ϵ_i's of a run form a learning curve. By averaging multiple runs one obtains the expected learning curve, $\bar{\epsilon}_i$.

The goal for the learner L is twofold. The error rates of the returned models ϵ_i's should (1) be as small as possible, and (2) be monotonically decreasing. These goals can be at odds with another. For example, always returning a fixed model ensures monotonicity but incurs large error rates. To measure (1), we summarize performance of a learning curve using the Area Under the Learning Curve (AULC) [12–14]. The AULC averages all ϵ_i's of a run. Low AULC indicates that a learner manages to quickly reduce the error rate.

Monotone in round i means that $\epsilon_{i+1} \leq \epsilon_i$. We may care about monotonicity of the expected learning curve *or* individual learning curves. In practice, however, we typically get one chance to gather data and submit models. In that case, we rather want to make sure that then any additional data also leads to better performance. Therefore, we are mainly concerned with monotonicity of *individual* learning curves. We quantify monotonicity of a run by the fraction of non-monotone transitions in an individual curve.

6.3. APPROACHES AND ALGORITHMS

We introduce three algorithms (learners L) that wrap around supervised learners with the aim of making them monotone. First, we provide some intuition how to achieve this: ideally, during the generation of the learning curve, we would check whether $\epsilon(h_{i+1}) \leq \epsilon(h_i)$. A fix to make a learner monotone would be to output h_i instead of h_{i+1} if the error rate of h_{i+1} is larger. Since learners do not have access to $\epsilon(h_i)$, we have to estimate it using the incoming data. The first two algorithms, $\text{MT}_{\text{SIMPLE}}$ and MT_{HT}, use the holdout method to this end; newly arriving data is partitioned into training and validation sets. The third algorithm, MT_{CV}, makes use of cross validation.

$\text{MT}_{\text{SIMPLE}}$: MONOTONE SIMPLE.

The pseudo-code for $\text{MT}_{\text{SIMPLE}}$ is given by Algorithm 2 in combination with the function UpdateSimple. Batches S^i are split into training (S^i_t) and validation (S^i_v). The training set S_t is enlarged each round with S^i_t and a new model h_i is trained. S^i_v is used to estimate the performance of h_i and h_{best}. We store the previously best performing model, h_{best}, and compare its performance to that of h_i. If the new model h_i is better, it is returned and h_{best} is updated, otherwise h_{best} is returned.

Algorithm 2: M_{SIMPLE} and M_{HT}

input : supervised learner A, rounds n, batches S^i
$u \in$ {updateSimple, updateHT}
if u =updateHT: confidence level α, hypothesis test HT

1 $S_t = \{\}$
2 **for** $i = 1, \ldots, n$ **do**
3 Split S^i in S_t^i and S_v^i
4 Append to $S_t : S_t = [S_t; S_t^i]$
5 $h_i \leftarrow A(S_t)$
6 $Update_i \leftarrow$ u(S_v^i, h_i, h_{best}, α, HT) `// see below`
7 Append to $S_t : S_t = [S_t; S_v^i]$
8 **if** $Update_i$ *or* $i = 1$ **then**
9 $h_{\text{best}} \leftarrow h_i$
10 **end**
11 Return h_{best} in round i
12 **end**

Function UpdateSimple

input : S_v^i, h_i, h_{best}

1 $P_{current} \leftarrow \hat{\epsilon}(h_i, S_v^i)$
2 $P_{best} \leftarrow \hat{\epsilon}(h_{\text{best}}, S_v^i)$
3 return $(P_{current} < P_{best})$

Function UpdateHT

input : S_v^i, h_i, h_{best}, confidence level α, hypothesis test HT

1 $p = HT(S_v^i, h_i, h_{\text{best}})$ `// p-value`
2 return $(p \leq alpha)$

Because h_i and h_{best} are both compared on S_v^i the comparison is more accurate because the comparison is paired. After the comparison S_v^i can safely be added to the training set (line 7 of Algorithm 2).

We call this algorithm MT_{SIMPLE} because the model selection is a bit naive: for small validation sets, the variance in the performance measure could be quite large, leading to many non-monotone decisions. In the limit of infinitely large S_v^i, however, this algorithm should always be monotone (and very data hungry).

MT_{HT}: MONOTONE HYPOTHESIS TEST.

The second algorithm, MT_{HT}, aims to resolve the issues of MT_{SIMPLE} with small validation set sizes. In addition, for this algorithm, we prove that individual learning curves are monotone with high probability. The same pseudo-code is used as for MT_{SIMPLE} (Alorithm 2), but with a different update function UpdateHT. Now a hypothesis test HT determines if the newly trained model is significantly better than the previous model. The hypothesis test makes sure that the newly trained model is not better due to chance (such as an unlucky sample). The hypothesis test is conservative, and only switches to a new model if we are reasonably sure it is significantly better, to avoid non-monotone decisions. Japkowicz and Shah [15] provide an accessible introduction to understand the frequentist hypothesis testing.

The choice of hypothesis test depends on the performance measure. For the error rate the McNemar test can be used [15, 16]. The hypothesis test should use paired data, since we evaluate two models on one sample, and it should be one-tailed. One-tailed, since we only want to know whether h_i is better than h_{best} (a two tailed test would switch to h_i if its performance is significantly different). The test compares two hypotheses: $H_0 : \epsilon(h_i) = \epsilon(h_{\text{best}})$ and $H_1 : \epsilon(h_i) < \epsilon(h_{\text{best}})$.

Several versions of the McNemar test can be used [15–17]. We use the McNemar exact conditional test which we briefly review. Let b be the random variable indicating the number of samples classified correctly by h_{best} and incorrectly by h_i of the sample S_v^i, and let N_d be the number of samples where they disagree. The test conditions on N_d. Assuming H_0 is true, $P(b = x|H_0, N_d) = \binom{N_d}{x}(\frac{1}{2})^{N_d}$. Given x b's, the p-value for our one tailed test is $p = \sum_{i=0}^{x} P(b = i|H_0, N_d)$.

The one tailed p-value is the probability of observing a more extreme sample given hypothesis H_0 considering the tail direction of H_1. The smaller the p-value, the more evidence we have for H_1. If the p-value is smaller than α, we accept H_1, and thus we update the model h_{best}. The smaller α, the more conservative the hypothesis test, and thus the smaller the chance that a wrong decision is made due to unlucky sampling. For the McNemar exact conditional test [17] the False Positive Rate (FPR, or the probability to make a Type I error) is bounded by α: $P(p \leq \alpha|H_0) \leq \alpha$. We need this to prove monotonicity with high probability.

MT_{CV}: MONOTONE CROSS VALIDATION.

In practice, often K-fold cross validation (CV) is used to estimate model performance instead of the holdout. This is what MT_{CV} does, and is similar to $\text{MT}_{\text{SIMPLE}}$. As described in Algorithm 3, for each incoming sample an index I maintains to which fold it belongs. These indices are used to generate the folds for the K-fold cross validation.

During CV, K models are trained and evaluated on the validation sets. We now have to memorize K previously best models, one for each fold. We average the performance of the newly trained models over the K-folds, and compare that to the average of the best previous K models. This averaging over folds is essential, as this reduces the variance of the model selection step as compared to selecting the best model overall (like $\text{MT}_{\text{SIMPLE}}$ does).

In our framework we return a single model in each iteration. We return the model with the optimal training set size that performed best during CV. This can further improve performance.

6.4. THEORETICAL ANALYSIS

We derive the probability of a monotone learning curve for $\text{MT}_{\text{SIMPLE}}$ and MT_{HT}, and we prove our algorithms are consistent if the model updates enough.

Theorem 11. *Assume we use the McNemar exact conditional test (see Section 6.3) with $\alpha \in (0, \frac{1}{2}]$, then the individual learning curve generated by Algorithm MT_{HT} with n rounds is monotone with probability at least $(1 - \alpha)^n$.*

Proof. First we argue that the probability of making a non-monotone decision in round i is at most α. If $H_1 : \epsilon(h_i) < \epsilon(h_{\text{best}})$ or $H_0 : \epsilon(h_i) = \epsilon(h_{\text{best}})$ is true, we are monotone in round i,

Algorithm 3: M_{CV}

input : K folds, learner A, rounds n, batches S^i

$b \leftarrow 1$ // keeps track of best round
$S = \{\}$, $I = \{\}$
for $i = 1, \ldots, n$ **do**
  Generate stratified CV indices for S^i and put in I^i. Each index i indicates to which validation fold the corresponding sample belongs.
  Append to S: $S \leftarrow [S; S^i]$
  Append to I: $I \leftarrow [I; I^i]$
  for $k = 1, \ldots, K$ **do**
    $h_i^k \leftarrow A(S[I \neq k])$ // training set of kth fold
    $P_i^k \leftarrow \hat{\epsilon}(h_i^k, S[I = k])$ // validation set of kth fold
    $P_b^k \leftarrow \hat{\epsilon}(h_b^k, S[I = k])$ // update performance of prev. models
  end
  $Update_i \leftarrow (mean(P_i^k) \leq mean(P_b^k))$ // mean w.r.t. k
  if $Update_i$ *or* $i = 1$ **then**
    $b \leftarrow i$
  end
  $k \leftarrow \arg\min_k P_b^k$ // break ties
  Return h_b^k in round i
end

so we only need to consider a new alternative hypothesis $H_2 : \epsilon(h_i) > \epsilon(h_{\text{best}})$. Under H_0 we have [17]: $P(p \leq \alpha | H_0) \leq \alpha$. Conditioned on H_2, b is binomial with larger mean than in the case of H_0, thus we observe larger p-values if $\alpha \in (0, \frac{1}{2}]$, thus $P(p \leq \alpha | H_2) \leq P(p \leq \alpha | H_0) \leq \alpha$. Therefore the probability of being non-monotone in round i is at most α. This holds for any model h_i, h_{best} and anything that happened before round i. Since S_v^i are independent samples, being non-monotone in each round can be seen as independent events, resulting in $(1-\alpha)^n$. □

If the probability of being non-monotone in all rounds is at most β, we can set $\alpha = 1 - \beta^{\frac{1}{n}}$ to fulfill this condition. Note that this analysis also holds for MT_{SIMPLE}, since running MT_{HT} with $\alpha = \frac{1}{2}$ results in the same algorithm as MT_{SIMPLE} for the McNemar exact conditional test.

We now argue that all proposed algorithms are consistent under some conditions. First, let us revisit the definition of consistency [3].

Definition 5 (Consistency[3])**.** *Let L be a learner that returns a hypothesis $L(S) \in H$ when evaluated on S. For all $\epsilon_{excess} \in (0,1)$, for all distributions P over $X \times Y$, for all $\delta \in (0,1)$, if there exists a $n(\epsilon_{excess}, P, \delta)$, such that for all $m \geq n(\epsilon_{excess}, P, \delta)$, if L uses a sample S of size m, and the following holds with probability (over the choice of S) at least $1 - \delta$,*

$$\epsilon(L(S)) \leq \min_{h \in H} \epsilon(h) + \epsilon_{excess}, \tag{6.2}$$

then L is said to be consistent.

Before we can state the main result, we have to introduce a bit of notation. U_i indicates the event that the algorithm updates h_{best} (or in case of M_{CV} it updates the variable b). H_i^{i+z} to indicates the event that $\neg U_i \cap \neg U_{i+1} \cap \ldots \cap \neg U_{i+z}$, or in words, that in round i to $i+z$ there has been no update. To fulfill consistency, we need that when the number of rounds grows to infinity, the probability of updating is large enough. Then consistency of A makes sure that h_{best} has sufficiently low error. For this analysis it is assumed that the number of rounds of the algorithms is not fixed.

Theorem 12. *MT_{SIMPLE}, MT_{HT} and MT_{CV} are consistent, if A is consistent and if for all i there exists a $z_i \in \mathbb{N}\backslash 0$ and $C_i > 0$ such that for all $k \in \mathbb{N}\backslash 0$ it holds that $P(H_i^{i+kz_i}) \leq (1-C_i)^k$.*

Proof. Let A be consistent with $n_A(\epsilon_{\text{excess}}, P, \delta)$ samples. Let us analyze round i where i is big enough such that[2] $|S_t| > n_A(\epsilon_{\text{excess}}, P, \frac{\delta}{2})$. Assume that

$$\epsilon(h_{\text{best}}) > \min_{h \in H} \epsilon(h) + \epsilon_{\text{excess}}, \tag{6.3}$$

otherwise the proof is trivial. For any round $j \geq i$, since A produces hypothesis h_j with $|S_t| > n_A(\epsilon_{\text{excess}}, P, \frac{\delta}{2})$ samples,

$$\epsilon(h_j) \leq \min_{h \in H} \epsilon(h) + \epsilon_{\text{excess}} \tag{6.4}$$

holds with probability of at least $1 - \frac{\delta}{2}$. Now L should update. The probability that in the next kz_i rounds we do not update is, by assumption, bounded by $(1-C_i)^k$. Since $C_i > 0$, we can choose k big enough so that $(1-C_i)^k \leq \frac{\delta}{2}$. Thus the probability of not updating after kz_i more rounds is at most $\frac{\delta}{2}$, and we have a probability of $\frac{\delta}{2}$ that the model after updating is not good enough. Applying the union bound we find the probability of failure is at most δ. □

A few remarks about the assumption. It tells us, that an update is more and more likely if we have more consecutive rounds where there has been no update. It holds if each z_i rounds the update probability is nonzero. A weaker but also sufficient assumption is $\forall_i$: $\lim_{z \to \infty} P(H_i^{i+z}) \to 0$.

For $\text{MT}_{\text{SIMPLE}}$ and MT_{CV} the assumption is always satisfied, because these algorithms look directly at the mean error rate—and due to fluctuations in the sampling there is always a non-zero probability that $\hat{\epsilon}(h_i) \leq \hat{\epsilon}(h_{\text{best}})$. However, for MT_{HT} this may not always be satisfied. Especially if the validation batches N_v are small, the hypothesis test may not be able to detect small differences in error—the test then has zero power. If N_v stays small, even in future rounds the power may stay zero, in which case the learner is not consistent.

6.5. EXPERIMENTS

We evaluate $\text{MT}_{\text{SIMPLE}}$ and MT_{HT} on artificial datasets to understand the influence of their parameters. Afterward we perform a benchmark where we also include MT_{CV} and

[2] In case of MT_{CV}, take $|S_t|$ to be the smallest training fold size in round i

a baseline that uses validation data to tune the regularization strength. This last experiment is also performed on the MNIST dataset to get an impression of the practicality of the proposed algorithms. First we describe the experimental setup in more detail.

EXPERIMENTAL SETUP.

The peaking dataset [4] and dipping dataset [6] are artificial datasets that cause non-monotone behaviour. We use stratified sampling to obtain batches S^i for the peaking and dipping dataset, for MNIST we use random sampling. For simplicity all batches have the same size. N indicates batch size, and N_v and N_t indicate the sizes of the validation and training sets.

As model we use least squares classification [18, 19]. This is ordinary linear least squares regression on the classification labels $\{-1, +1\}$ with intercept. For MNIST one-versus-all is used to train a multi-class model. In case there are less samples for training than dimensions, the required inverse of the covariance matrix is ill-defined and we resort to the Moore-Penrose Pseudo-Inverse.

Monotonicity is calculated by the fraction of non-monotone iterations per run. AULC is also calculated per run. We do 100 runs with different batches and average to reduce variation from the randomness in the batches. Each run uses a newly sampled test set consisting of 10000 samples. The test set is used to estimate the true error rate and is not accessible by any of the algorithms. Because of the large number of runs the standard errors of the learning curve are quite small and therefore we do not display them to improve the clarity of the figures.

We evaluate $\mathrm{M}_{\mathrm{SIMPLE}}$, M_{HT} and M_{CV} and several baselines. The standard learner just trains on all received data. A second baseline, λ_S, splits the data in train and validation like $\mathrm{M}_{\mathrm{SIMPLE}}$ and uses the validation data to select the optimal L_2 regularization parameter λ for the least square classifier. Regularization is implemented by adding λI to the estimate of the covariance matrix.

In the first experiment we investigate the influence of N_v and α for $\mathrm{MT}_{\mathrm{SIMPLE}}$ and $\mathrm{MT}_{\mathrm{HT}}$ on the decisions. A complicating factor is that if N_v changes, not only decisions change, but also training set sizes because S_v is appended to the training set (see line 7 of Algorithm 2). This makes interpretation of the results difficult because decisions are then made in a different context. Therefore, for the first set of experiments, we do not add S_v to the training sets, also not for the standard learner. For this set of experiment We use $N_t = 4$, $n = 150$, $d = 200$ for the peaking dataset, and we vary α and N_v.

For the benchmark, we set $N_t = 10$, $N_v = 40$, $n = 150$ for peaking and dipping, and we set $N_t = 5$, $N_v = 20$, $n = 40$ for MNIST. We fix $\alpha = 0.05$ and use $d = 500$ for the peaking dataset. For MNIST, as preprocessing step we extract 500 random Fourier-features as also done by Belkin et al. [7]. For $\mathrm{MT}_{\mathrm{CV}}$ we use $K = 5$ folds. For λ_S we try $\lambda \in \{10^{-5}, 10^{-4.5}, \ldots, 10^{4.5}, 10^5\}$ for peaking and dipping, and we try $\lambda \in \{10^{-3}, 10^{-2}, \ldots, 10^3\}$ for MNIST.

RESULTS.

We perform a preliminary investigation of the algorithms $\mathrm{M}_{\mathrm{SIMPLE}}$ and M_{HT} and the influence of the parameters N_v and α. We show several learning curves in Figure 6.1a and 6.1d. For small N_v and α we observe $\mathrm{MT}_{\mathrm{HT}}$ gets stuck: it does not switch models anymore, indicating that consistency could be violated.

In Figure 6.1b and Figure 6.1e we give a more complete picture of all tried hyperparameters in terms of the AULC. In Figure 6.1c and Figure 6.1f we plot the fraction of non-monotone decisions during a run (note that the legends for the subfigures are different). Observe that the axes are scaled differently (some are logarithmic). In some cases zero non-monotone decisions were observed, resulting in a missing value due to log(0). This occurs for example if $\mathrm{MT}_{\mathrm{HT}}$ always sticks to the same model, then no non-monotone decisions are made. The results of the benchmark are shown in Figure 6.2. The AULC and fraction of monotone decisions are given in Table 6.1.

6.6. DISCUSSION

FIRST EXPERIMENT: TUNING α AND N_v.

As predicted $\mathrm{MT}_{\mathrm{SIMPLE}}$ typically performs worse than $\mathrm{MT}_{\mathrm{HT}}$ in terms of AULC and monotonicity unless N_v is very large. The variance in the estimate of the error rates on S_v^i is so large that in most cases the algorithm does not switch to the correct model. However, $\mathrm{MT}_{\mathrm{SIMPLE}}$ seems to be consistently better than the standard learner in terms of monotonicity and AULC, while $\mathrm{MT}_{\mathrm{HT}}$ can perform worse if badly tuned.

Larger N_v leads typically to improved AULC for both. $\alpha \in [0.05, 0.1]$ seems to work best in terms of AULC for most values of N_v. If α is too small, $\mathrm{MT}_{\mathrm{HT}}$ can get stuck, if α is too large, it switches models too often and non-monotone behaviour occurs. If $\alpha \to \frac{1}{2}$, $\mathrm{MT}_{\mathrm{HT}}$ becomes increasingly similar to $\mathrm{MT}_{\mathrm{SIMPLE}}$ as predicted by the theory.

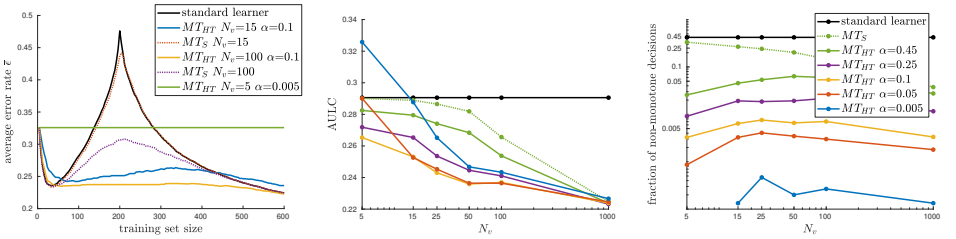
The fraction of non-monotone decisions of $\mathrm{MT}_{\mathrm{HT}}$ is much lower than α. This is in agreement with Theorem 11, but could indicate in addition that the hypothesis test is rather pessimistic. The standard learner and $\mathrm{MT}_{\mathrm{SIMPLE}}$ often make non-monotone decisions. In some cases almost 50% of the decisions are not-monotone.

SECOND EXPERIMENT: BENCHMARK ON PEAKING, DIPPING, MNIST.

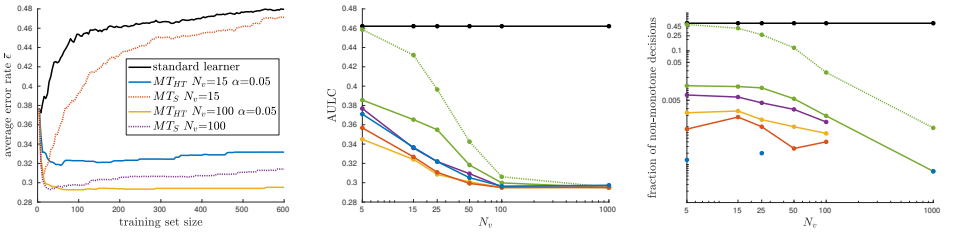
Interestingly, for peaking and MNIST datasets any non-monotonicity (double descent [7]) in the *expected* learning curve almost completely disappears for λ_S, which tunes the regularization parameter using validation data (Figure 6.2). We wonder if regularization can also help reducing the severity of double descent in other settings. For the dipping dataset, regularization does not help, showing that it cannot prevent non-monotone behaviour. Furthermore, the fraction of non-monotone decisions *per run* is largest for this learner (Table 6.1).

For the dipping dataset M_{CV} has a large advantage in terms of AULC. We hypothesize that this is largely due to tie breaking and small training set sizes due to the 5-folds. Surprisingly on the peaking dataset it seems to learn quite slowly. The expected learning curves of $\mathrm{MT}_{\mathrm{HT}}$ look better than that of $\mathrm{MT}_{\mathrm{SIMPLE}}$, however, in terms of AULC the difference is quite small.

The fraction of non-monotone decisions for $\mathrm{MT}_{\mathrm{HT}}$ per run is very small as guaranteed. However, it is interesting to note that this does not always translate to monotonicity in the expected learning curve. For example, for peaking and dipping the expected curve does not seem entirely monotone. But $\mathrm{MT}_{\mathrm{CV}}$, which makes many non-monotone decisions per run, still seems to have a monotone expected learning curve. While monotonicity of each

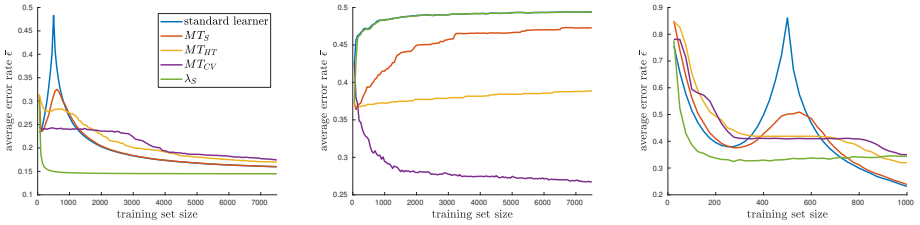


(a) Peaking learning curve (b) Peaking AULC (c) Peaking Monotonicity



(d) Dipping learning curve (e) Dipping AULC (f) Dipping Monotonicity

Figure 6.1.: Influence of N_v and α for $\text{MT}_{\text{SIMPLE}}$ and MT_{HT} on the Peaking and Dipping dataset. Note that some axes are logarithmic and b, c, e, f have the same legend.



(a) Peaking (b) Dipping (c) MNIST

Figure 6.2.: Expected learning curves on the benchmark datasets.

individual learning curves guarantees monotonicity in the expected curve, this result indicates monotonicity of each individual curve may not be necessary. This raises the question: under what conditions do we have monotonicity of the expected learning curve?

GENERAL REMARKS.

The fraction of non-monotone decisions of MT_{HT} being so much smaller than α could indicate the hypothesis test is too pessimistic. Fagerland et al. [17] note that the asymptotic McNemar test can have more power, which could further improve the AULC. For this test the guarantee $P(p \leq \alpha | H_0) \leq \alpha$ can be violated, but in light of the monotonicity results obtained, practically this may not be an issue.

MT_{HT} is inconsistent at times, but this does not have to be problematic. If one knows

Table 6.1.: Results of the benchmark. SL is the Standard Learner. AULC is the Area Under the Learning Curve of the error rate. Fraction indicates the average fraction of non-monotone decisions during a single run. Standard deviation shown in (braces). Best monotonicity result is underlined.

	AULC Peaking	AULC Dipping	AULC MNIST
SL	0.198 (0.003)	0.49 (0.01)	0.44 (0.01)
MT_S	0.195 (0.005)	0.45 (0.06)	0.42 (0.02)
MT_{HT}	<u>0.208</u> (0.009)	<u>0.38</u> (0.08)	<u>0.45</u> (0.02)
MT_{CV}	0.208 (0.005)	0.28 (0.02)	0.45 (0.01)
λ_S	0.147 (0.003)	0.49 (0.01)	0.36 (0.02)

	Fraction Peaking	Fraction Dipping	Fraction MNIST
SL	0.31 (0.02)	0.50 (0.03)	0.27 (0.04)
MT_S	0.23 (0.03)	0.37 (0.15)	0.11 (0.04)
MT_{HT}	<u>0.00</u> (0.00)	<u>0.00</u> (0.00)	<u>0.00</u> (0.00)
MT_{CV}	0.34 (0.03)	0.19 (0.08)	0.30 (0.06)
λ_S	0.43 (0.03)	0.50 (0.03)	0.46 (0.05)

the desired error rate, a minimum N_v can be determined that ensures the hypothesis test will not get stuck before reaching that error rate. Another possibility is to make the size N_v dependent on i: if N_v is monotonically increasing this directly leads to consistency of $\mathrm{MT_{HT}}$. It would be ideal if somehow N_v could be automatically tuned to trade off sample size requirements, consistency and monotonicity. Since for CV N_v automatically grows and thus also directly implies consistency, a combination of $\mathrm{MT_{HT}}$ and $\mathrm{MT_{CV}}$ is another option.

Devroye et al. [20] conjectured that it is impossible to construct a consistent learner that is monotone in terms of the expected learning curve. Since we look at individual curves, our work does not disprove this conjecture, but some of the authors on this paper believe that the conjecture can be disproved. One step to make is to get to an essentially better understanding of the relation between individual learning curves and the expected one.

Currently, our definition judges any decision that increases the error rate, by however small amount, as non-monotone. It would be desirable to have a broader definition of non-monotonicity that allows for small and negligible increases of the error rate. Using a hypothesis test satisfying such a less strict condition could allow us to use less data for validation.

Finally, the user of the learning system should be notified that non-monotonicity has occurred. Then the cause can be investigated and mitigated by regularization, model selection, etc. However, in automated systems our algorithm can prevent any known and unknown causes of non-monotonicity (as long as data is i.i.d.), and thus can be used as a failsafe that requires no human intervention.

6.7. CONCLUSION

We have introduced three algorithms to make learners more monotone. We proved under which conditions the algorithms are consistent and we have shown for $\mathrm{MT}_{\mathrm{HT}}$ that the learning curve is monotone with high probability. If one cares only about monotonicity of the expected learning curve, $\mathrm{MT}_{\mathrm{SIMPLE}}$ with very large N_v or $\mathrm{MT}_{\mathrm{CV}}$ may prove sufficient as shown by our experiments. If N_v is small, or one desires that individual learning curves are monotone with high probability (as practically most relevant), $\mathrm{MT}_{\mathrm{HT}}$ is the right choice. Our algorithms are a first step towards developing learners that, given more data, improve their performance in expectation.

6.8. BIBLIOGRAPHY

[1] T. J. Viering, A. Mey, and M. Loog, *Making learners (more) monotone,* in *International Symposium on Intelligent Data Analysis* (Springer, 2020) pp. 535–547.

[2] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning* (MIT Press, 2012).

[3] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms* (Cambridge university press, USA, 2014).

[4] R. Duin, *Small sample size generalization,* in *Proceedings of the Scandinavian Conference on Image Analysis*, Vol. 2 (1995) pp. 957–964.

[5] M. Opper, W. Kinzel, J. Kleinz, and R. Nehl, *On the ability of the optimal perceptron to generalise,* Journal of Physics A: Mathematical and General **23**, L581 (1990).

[6] M. Loog and R. Duin, *The dipping phenomenon,* in *S+SSPR* (Hiroshima, Japan, 2012) pp. 310–317.

[7] M. Belkin, D. Hsu, S. Ma, and S. Mandal, *Reconciling modern machine-learning practice and the classical biasvariance trade-off,* Proceedings of the National Academy of Sciences **116**, 15849 (2019).

[8] S. Spigler, M. Geiger, S. D'Ascoli, L. Sagun, G. Biroli, and M. Wyart, *A jamming transition from under- to over-parametrization affects loss landscape and generalization,* arXiv preprint arXiv:1810.09665 (2018).

[9] T. Viering, A. Mey, and M. Loog, *Open problem: Monotonicity of learning,* in *Conference on Learning Theory, COLT* (2019) pp. 3198–3201.

[10] M. Loog, T. Viering, and A. Mey, *Minimizers of the Empirical Risk and Risk Monotonicity,* in *NeuRIPS 32* (2019) pp. 7476–7485.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition,* Proceedings of the IEEE **86**, 2278 (1998).

[12] J. ONeill, S. J. Delany, and B. MacNamee, *Model-free and model-based active learning for regression,* in *16th UK Workshop on Computational Intelligence* (Springer, 2016) pp. 375–386.

[13] M. Huijser and J. C. van Gemert, *Active decision boundary annotation with deep generative models,* in *ICCV* (2017) pp. 5286–5295.

[14] B. Settles and M. Craven, *An analysis of active learning strategies for sequence labeling tasks,* in *EMNLP* (2008) pp. 1070–1079.

[15] N. Japkowicz and M. Shah, *Evaluating Learning Algorithms: A Classification Perspective* (Cambridge University Press, 2011).

[16] S. Raschka, *Model evaluation, model selection, and algorithm selection in machine learning,* arXiv preprint arXiv:1811.12808 (2018).

[17] M. W. Fagerland, S. Lydersen, and P. Laake, *The McNemar test for binary matched-pairs data: mid-p and asymptotic are better than exact conditional.* BMC medical research methodology **13**, 91 (2013).

[18] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning* (Springer, 2009).

[19] R. Rifkin, G. Yeo, and T. Poggio, *Regularized least-squares classification,* Nato Science Series Sub Series III Computer and Systems Sciences **190**, 131 (2003).

[20] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Stochastic Modelling and Applied Probability (Springer, 1996).

7

DISCUSSION

We have discussed the concept of safety with respect to three topics: explainability, active learning and supervised learning. We illustrated that Grad-CAM as an explanation method can be unsafe for neural networks that otherwise perform well. In active learning, we showed that tighter generalization bounds do not imply that one algorithm is better than the other. Instead, we should take into account the probability of the worst-case scenario that such bounds consider. An average case analysis gave us a more accurate picture of the performance of different approaches.

In the context of learning curves we have come back to the foundation of machine learning and studied whether supervised learning itself is safe, in the sense that, whether more data leads to improvement in terms of the expected risk. The review has provided an overview regarding known results of non-monotone and monotone learning curves, mostly illustrating that for very little settings monotonicity results are known, and that plenty of non-monotone examples exist. Even non-monotonicity can occur in benign settings where there is no model misspecification, indicating perhaps that monotonicity is indeed a very difficult property to satisfy. This is corroborated by the fact that we found more novel examples that show non-monotone behavior, even in simple settings. However, we end on a positive note, as the last chapter illustrates that it is possible to make learning curves somewhat more monotone using wrapper algorithms.

We now discuss what we see as the three most interesting directions for further research as inspired by this work, in the context of the three topics: explainability, active learning and learning curves. Afterward, we wrap up with a discussion with a final reflection on our concept of safety in machine learning.

7.1. MONOTONE IN EXPECTATION?

In Chapter 6 we have discussed how to make learning curves monotone with high probability with wrapper algorithms. In the experiments we have also observed that the learning curves of the wrapper became somewhat more monotone in expectation. Up until now it has remained impossible for us to prove anything regarding monotonicity of the expected learning curve of the wrapper algorithm.

We think this an intriguing issue that deserves further study, because, this in turn could

prove or falsify the conjecture of Devroye et al. [1]. They conjectured that it is impossible to have learners that are monotone in expectation and that are universally consistent. Universally consistent means a learner converges to the Bayes optimal risk when the amount of data grows to infinity for all possible distributions $P_{\mathcal{X}\mathcal{Y}}$. Applying the technique of the wrapper algorithm to universally consistent learners should result in learners who are also monotone with high probability. If we would know more about when the expected curve of wrappers would become monotone in expectation, this knowledge could then help prove or disprove the conjecture. One technical issue is that the proof of monotonicity with high probability relies on the fact that there are a finite amount of rounds. However, we think it is possible to extend the analysis to infinite rounds for the MT_{HT} wrapper algorithm, by making the confidence level α of the hypothesis test dependent on the round.

One way to study the question of whether monotonicity in expectation is achievable by wrapper algorithms, is to assume a distribution on the learning curve of the original learner $A(S)$. We have come up with a problem whose learning curve distribution is known, and where we conjecture that the wrapper algorithm $\text{MT}_{\text{SIMPLE}}$ can make the expected learning curve *strictly* monotone. We see this as a first step towards understanding when it is possible to make the curve monotone in expectation. The example is as follows.

Example 1. *Assume we have a 1D classification problem, where $p(x)$ is the uniform distribution between -0.5 and 0.5. Assume that the ground truth decision rule is linear and that there is no model misspecification. Without loss of generality we can assume that the true decision rule is $sign(x)$. We choose as hypothesis class all possible linear models on the interval $[-0.5, 0.5]$, which we can parameterize as $H = \{h(x) = sign(xw+b); w \in \{-1, 1\}, b \in [-0.5, 0.5]\}$. Let $A(S)$ be the following peculiar learner. Each time it is invoked, it ignores the sample S, and selects uniformly a random model $h \in H$. For example, it could determine w by a fair coinflip and sample b randomly from a uniform distribution on $[-0.5, 0.5]$.*

For this example, we can analytically compute the error rate for each model h. We find that the error rate is given by $|b|$ for $w = 1$, and $1 - |b|$ for $w = -1$, see Figure 7.1. Note that

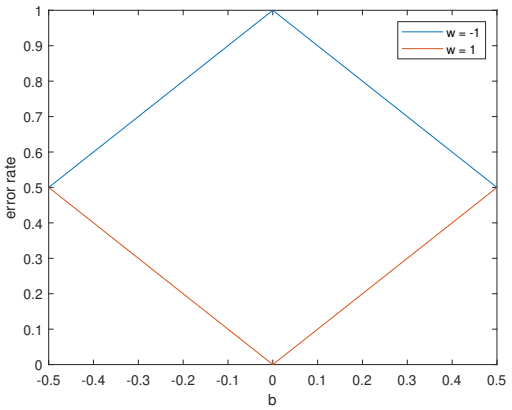


Figure 7.1.: The error rate depending on the parameters of h.

the error rate for $w = 1$ is in $[0, 1/2]$ as here the model has the correct sign, while for $w = -1$ the error rate is in $[1/2, 1]$. Going through the motions, we can compute the distribution of $R(A(S))$, e.g. the distribution of the error rate of the model returned by $A(S)$. However, from the fact that the distribution b is uniform, and that w is also uniform on $\{-1, +1\}$, and from inspection of Figure 7.1, we can conclude that the error rate $R(A(S))$ must also be a uniform random variable on $[0, 1]$ — due to symmetry considerations this must be the case. Unlike usual learning algorithms, the distribution of the error rate does not depend on n.

This gives us thus directly the distribution of the learning curve, see Figure 7.2 (left) for an illustration. The mean of the error rate is 0.5 for each sample size, and therefore the expected learning curve is constant. In view of monotonicity, the expected curve is at least monotone, but not strictly. Note that a non-averaged learning curve on the other hand is extremely ill-behaved, as the consecutive error rates are i.i.d. random samples from $[0, 1]$, and thus would be highly non-monotone.

For this case it is possible to compute the distribution of the learning curve of the wrapper algorithm in closed form for $\text{MT}_{\text{SIMPLE}}$ in case $N_v = 1$, that is, where a validation set of a single sample is used. A short derivation is given in Appendix E. The distribution is illustrated in Figure 7.2 (right) for a few sample sizes. It can be seen that the expected curve behaves strictly monotone, but it remains an open issue to prove this rigorously for all sample sizes. If we could prove it, it would be the first example where we show that the wrapper algorithm can make the expected curve strictly monotone.

This example is of course unrealistic, as the error rate for the model would never be uniform on $[0, 1]$. But we can trivially also plug in other distributions to model the error rate in this exact computation. For example, we could model the error rate as being uniform between $[a(n), b(n)]$, for $0 < a(n) < b(n) < 1$, where $a(n)$ and $b(n)$ are dependent on the sample size n. This may already give a reasonable model to approximate error rates for a model trained on actual data. If we have a good estimate of these distributions, we can use the same procedure as described in Appendix E to compute the exact learning curve of the wrapper algorithm for these more realistic settings.

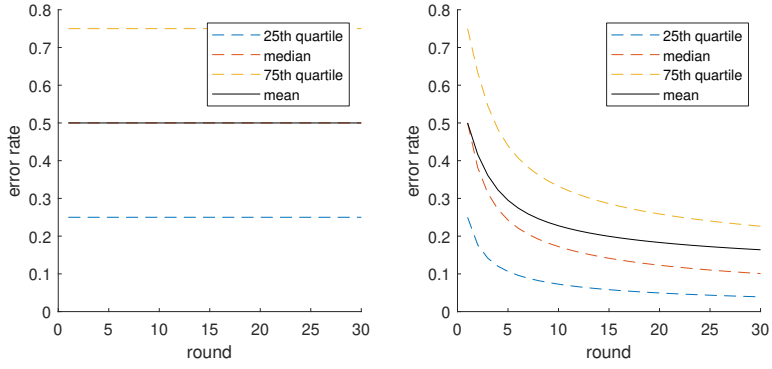


Figure 7.2.: In the left subfigure the distribution of the learning curve is given for $A(S)$, in the right subfigure the distribution of the wrapper algorithm $\text{MT}_{\text{SIMPLE}}$ is shown.

One issue is that in this appendix we have assumed that the error rate of the models with different sample sizes are independent. This may not be the case in practice when a training set is enlarged iteratively when building the learning curve, e.g. using monotonically increasing training sets. To illustrate, let us assume the existence of an outlier object which confuses the model during training, impacting performance negatively. If the training set S_n has as nth sample this outlier, the next training set S_{n+1} will also contain this outlier, and thus their performances are dependent. However, at this point, it is not clear whether including these dependencies is necessary to analyze the expected learning curve.

Initially, we would suggest to plug in hand-crafted learning curve distributions to get a better understanding of the wrapper behavior. For some learning curve distributions it should be clear that wrappers can never make the expected curve monotone. For example, if the distribution of the error rate conditioned on n samples is always smaller than the distribution of the error rate conditioned on $n+1$ samples, the expected curve can never be monotone if the wrapper has a non-zero probability of switching to the model with $n+1$ samples (see Figure 7.3, left). Typically, we would expect that the wrapper has a non-zero probability of switching in every round, as this seems necessary to satisfy universal consistency.

We anticipate a few general rules that govern whether the expected curve can be made monotone. Inspired by the previous discussion, one condition is that the distribution of the error rate for different sample sizes should have sufficient overlap (see Figure 7.3, right). By the same logic, when looking for learners that can be made monotone by the wrapper, we should look for learners $A(S)$ that do not learn too fast. Learning too fast reduces the overlap between consecutive sample sizes and thus may lead to unavoidable non-monotonicity if the curve is going up. As another option, we may consider injecting noise into $A(S)$, as this may cause the spread of the error rate to increase, leading to more overlap, thus making it more feasible to make the expected curve monotone even if the error rate of $A(S)$ is going

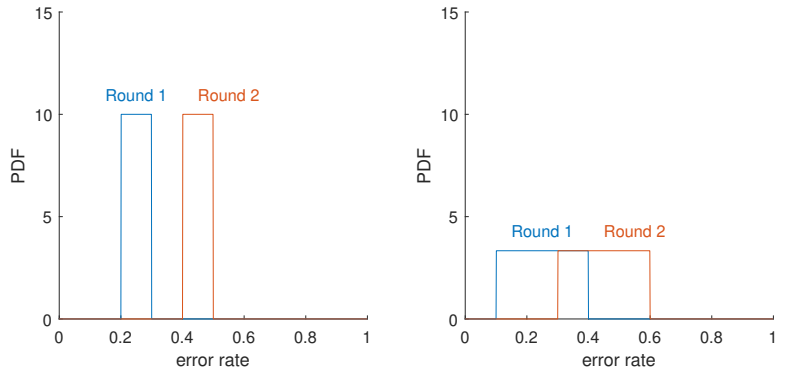


Figure 7.3.: Different learning curve distributions of $A(S)$. Left: this learning problem cannot be made monotone by wrapper algorithms. Right: by increasing the spread of both distributions, overlap is increased, perhaps making it possible to monotonify the expected curve of the wrapper algorithm even if the error rate is increasing.

up. Furthermore, we should also avoid learners $A(S)$ that reach the best possible error rate for finite sample sizes. Because if this occurs, the error can only go up, and we may obtain non-monotonicity.

Besides that, we can study different wrapper algorithms and the influence of their hyper-parameters. In particular, it would be interesting to investigate MT_{HT}, as it may be able to achieve monotonicity with high probability for an infinite amount of rounds. Ideally, we would study the class of all wrapper algorithms, as this will tell us more generally when it is possible to achieve monotonicity of the expected curve. A more in depth investigation of these issues will lead us to a better understanding of when monotonicity of the expected curve is possible, and perhaps will help to settle whether the conjecture raised by Devroye et. al. is true or not.

While preparing this dissertation, two new works make further steps into resolving monotonicity in expectation. A first step was made by Mhammedi [2], who initially claimed to have developed a wrapper algorithm that makes the learning curve monotone in expectation. Just like our wrapper algorithm, it relies on switching to a new model if there is sufficient confidence that the new model is better. Using the fact that losses are bounded, the confidence can be tuned in such a way that the loss in expectation is bounded. However, as pointed out by Bousquet *et al.* [3], there was a small mistake in the proof for the expected curve, and it turns out that the expectation case can still show some small bounded non-monotonicity.

Following similar arguments as we gave just above, Bousquet *et al.* [3] realize in a preprint that in some cases a wrapper algorithm cannot make the curve monotone. In particular, they consider the case where the learner is deterministic, and thus the learning curve is a series of delta peaks. In that case, there is zero overlap between distributions of the learning curve, and the curve cannot be made monotone by a wrapper algorithm, as we have argued before. One way to get around this impossibility result, is to degrade the performance of learners. By an intelligent scheme where older learners are slightly degraded in their performance by adding noise, Bousquet *et al.* [3] prove monotonicity in expectation. This is done in a similar way as our wrapper algorithm, by checking the performance of competing hypotheses on a validation set, and accepting new models only if they perform better with sufficiently large margin. Their analysis focuses on the case of the zero-one loss for multi-class and binary classification. They argue that by applying their technique to universally consistent learners, that they have resolved the conjecture of Devroye et al. Interestingly, Pestov [4] actually resolved the conjecture of Devroye et al. just a year earlier. However, their focus is on a specific histogram learner. Meanwhile, the analysis of Bousquet *et al.* [3], is applicable to any kind of learner, in the same spirit as our wrapper algorithm. Besides a rigorous checking of these theoretical results, it remains to be seen how practical these proposed algorithms are.

Furthermore, Mhammedi [2] uses the training set to make decisions. Because of overfitting concerns, necessarily this approach must be pessimistic. In case of learners such as the 1NN, whose training error will always be zero, it remains to be seen how useful this algorithm is. We believe that for these reasons, it may still be more practical to split data in training and validation sets, and to use validation data to make decisions to make the curve monotone. Ideally we would characterize the sample efficiency and degree of monotonicity of both approaches to find out which approach is to be preferred in what case.

7.2. SAFE EXPLANATION METHODS

Regarding explainability, Chapter 2 shows that Grad-CAM is not robust to manipulations of neural networks by an adversary that do not affect the performance significantly. One shortcoming of our attack was that the network architecture had to be changed, as this makes the scenario more pathological. It also makes the attack easily recognizable in practical settings since changes in architecture should be apparent in code.

However, recent work has shown that an attack in the same spirit can be carried out by only changing the values of the network weights [5], proving that such attacks are not pathological and indeed are realistic. The practical take home messages here are that one should be careful to check that weights are not manipulated by an attacker when downloading them. Moreover, when neural networks are used in high stakes settings, it would be good to check that weights are not manipulated from time to time. Both can be accomplished using cryptographic hashing, such as using an MD5 or, even better, a SHA checksum [6].

On the more theoretical side, that networks exist that can mislead explenations is problematic. While it seems unlikely that networks trained with regular loss functions (cross entropy, etc.) exhibit such misleading explanations, it would be very comforting to somehow prove when the explanation methods are safe to use. Because we currently do not fully understand how deep neural network training works, it seems impossible to rule out that by accident gradient descent will construct a network with misleading explanations. Ideally, we would have a condition that can be checked, to assure us whether the explanations will be faithful.

Since Grad-CAM uses the gradient, perhaps misleading explanations can be detected by looking at the magnitude of second order or higher order derivatives of the network output with respect to the input. For example, if second order derivatives are very large close to the object that is being explained, it indicates that the gradient may not be trusted as small changes in pixel values could lead to large changes in the gradient. Perhaps regularization of such higher order derivatives during training may guarantee trustworthy explanation of the final model? If one really has to deal with an adversary that provides a network and if the networks are sufficiently flexible, even higher order derivatives may turn out to be unhelpful, since the network could achieve misleading explanations using discontinuities. In that case, derivatives computed using backpropegation cannot help us, and we will need to resort to approximating derivatives using the method of finite differences. In other words, we will need to perform multiple forward passes on perturbed versions of the input to determine derivatives, which could be quite time consuming.

Clearly, ruling out backdoors like these efficiently in a neural network is a non-trivial task. Constructing more counterexamples that mislead explanation-methods seems like a fruitful avenue for future work. Indeed, fairly recently while refining this manuscript, Fokkema *et al.* [7] prove that it is impossible to be robust and sensitive for a wide range of refined explanation methods, where sensitivity relates to the sensitivity of detecting the correct explanation. This may show us the fundamental limit of what is explainable.

7.3. SAFE ACTIVE LEARNING

Chapter 3 has illustrated that tighter generalization bounds do not necessarily imply better active learners. One important limitation of our work is that it remains unclear when the

active learners can beat random sampling, as this would be the logical baseline to compare an active learner to. If we assume that the groundtruth model is sampled from a prior distribution, we can perform simulations to measure the performance of the active learners compared to random sampling. This way we can predict beforehand the optimal algorithm to use. In principle, this approach can be used to build a safe active learner that never performs worse than random sampling. However, this crucially relies on knowing the prior. Generally, we expect that so little is known about the prior distribution for general learning problems, that any such simulation will be practically useless.

In a more general setting, where no prior or problem average is assumed, it remains an open problem to show whether a more practical approach to safety is possible in active learning. In the spirit of the wrapper algorithm that we used to make the learning curve more monotone, we may also consider wrapper algorithms for active learning. For example, we could sample 40% of the budget randomly, we could sample 40% actively using any particular strategy, and use both datasets to construct two classifiers. The remaining 20% could be used to perform a hypothesis test that decides which model to use in the end. This construction should at least always be competitive with random sampling on 40% on all of the data (with high probability), even if the learning curve of random sampling or the particular active learning strategy are non-monotone. On the other hand, we may argue that if we always choose either strategy, 40% of the labeling budget is always wasted.

Wrapper algorithms for active learning have in fact already been proposed before [8–11]. Typically, such wrapper algorithms use the multi-armed bandit framework [12] to decide which strategy to use for each round. Such bandit algorithms use feedback, called rewards, after each round, which helps them decide which active learning strategy is promising. As such, already before exhausting the labeling budget, feedback is incorporated to select the most promising active learning strategy, and thus this may alleviate the concern of wasteful labeling somewhat.

However, most aforementioned works take the importance-weighted accuracy on the training set as reward function for the bandit [9–11]. Therefore, learners that have zero training error, such as 1-NN, cannot be guided to the correct sampling strategy. Similarly to the design of our wrapper algorithm for supervised learners, we therefore believe it would be fruitful to design such a bandit algorithm where feedback is coming from a validation set. We believe such steps could lead to active learners with stricter safety guarantees.

7.4. FINAL WORDS ON SAFETY

The concept that we call safety differs from what is usually studied in statistical learning theory, and can offer different insights. In statistical learning theory, usually the performance of the algorithm is compared to the best model from the class, resulting in the quantity called excess risk. Generalization bounds are derived on the excess risk which give theoretical guarantees. In both cases, a performance comparison is made to an ideal model that we can never attain. In contrast, for our proposed notion of safety, we compare the performance of one algorithm with a natural baseline, where both are algorithms which can actually be applied. As such, safety guarantees seem more practically relevant, than the guarantees usually provided by learning theory.

As we have seen in active learning, tighter bounds on the excess risk do not imply that

one algorithm always beats the other (safety). That we know relatively little about safety should be clear, but some first steps have now be taken to develop learners with monotone learning curves. But many questions remain; about their efficiency, their optimal design, and what kind of theoretical guarantees are possible. The results regarding semi-supervised learning and domain adaptation where safety was possible in the transductive setting seem surprisingly strong. Perhaps revisiting the transductive setting could be therefore be fruitful to study the monotonicity of supervised learning.

Furthermore, questions regarding safety can be raised in various settings, such as explainability, but perhaps also in the context of fairness, privacy, and others. We see the topic of safety as an important research direction, as it can tell us when which algorithm should be preferred and why. Hopefully, continued study of this topic will lead to a better understanding of when we can safely use our algorithms and when we should instead rely on simpler baselines.

7.5. BIBLIOGRAPHY

[1] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Vol. 31 (Springer, New York, NY, USA, 1996).

[2] Z. Mhammedi, *Risk monotonicity in statistical learning,* Advances in Neural Information Processing Systems **34** (2021).

[3] O. Bousquet, A. Daniely, H. Kaplan, Y. Mansour, S. Moran, and U. Stemmer, *Monotone learning,* arXiv preprint arXiv:2202.05246 (2022).

[4] V. Pestov, *A universally consistent learning rule with a universally monotone error,* arXiv preprint arXiv:2108.09733 (2021).

[5] J. Heo, S. Joo, and T. Moon, *Fooling neural network interpretations via adversarial model manipulation,* in *Advances in Neural Information Processing Systems* (2019) pp. 2925–2936.

[6] J. C. Lubbe, *Basic methods of cryptography* (Cambridge University Press, 1998).

[7] H. Fokkema, R. de Heide, and T. van Erven, *Attribution-based explanations that provide recourse cannot be robust,* arXiv preprint arXiv:2205.15834 (2022).

[8] Y. Baram, R. E. Yaniv, and K. Luz, *Online choice of active learning algorithms,* Journal of Machine Learning Research **5**, 255 (2004).

[9] W.-N. Hsu and H.-T. Lin, *Active learning by learning,* in *Twenty-Ninth AAAI conference on artificial intelligence* (Citeseer, 2015).

[10] K. Pang, M. Dong, Y. Wu, and T. M. Hospedales, *Dynamic ensemble active learning: A non-stationary bandit with expert advice,* in *2018 24th International Conference on Pattern Recognition (ICPR)* (IEEE, 2018) pp. 2269–2276.

[11] H.-M. Chu and H.-T. Lin, *Can active learning experience be transferred?* in *2016 IEEE 16th International Conference on Data Mining (ICDM)* (IEEE, 2016) pp. 841–846.

[12] S. Bubeck and N. Cesa-Bianchi, *Regret analysis of stochastic and nonstochastic multi-armed bandit problems,* Machine Learning **5**, 1 (2012).

A

Open Problem: Monotonicity of Learning

We pose the question to what extent a learning algorithm behaves monotonically in the following sense: does it perform better, in expectation, when adding one instance to the training set? We focus on empirical risk minimization and illustrate this property with several examples, two where it does hold and two where it does not. We also relate it to the notion of PAC-learnability.

This work was presented as an open problem at COLT 2019 and has been published as an extended abstract in Volume 99 of the Proceedings of Machine Learning Research [1].

A

A.1. INTRODUCTION

Recently, there has been an increasing amount of attention on machine learning algorithms that are presently referred to as robust or safe, meaning that even when assumptions are violated, performance will not degrade significantly [2]. The focus is mostly on settings that are slightly different from supervised learning such as online learning [3], domain adaptation [4] and semi-supervised learning [5]. The open problem presented here makes the point that such robustness and safety properties are not even fully understood for standard supervised learning and density estimation.

We focus on what we will refer to as the *monotonicity* of a learner's performance: given one additional training instance, to what extent can we expect a learner to improve? Or, equivalently, when is the learning curve monotone [6]? While this property is undoubtedly desirable, and most of us expect such behavior, there are surprising counterexamples. This open problem asks to unravel this behavior.

Understanding theoretical properties of learning curves can set expectations for practitioners. For example, if we know that a learner is monotone, but we observe the opposite in practice, we know that this behaviour must have another explanation, such as a finite sampling effect.

A.2. PRELIMINARIES AND RELATED WORK

Let $S_n = (z_1, \ldots, z_n)$ be a training set of size n, sampled i.i.d. from an (unknown) distribution D over a domain $\mathcal{Z}$. The learner A we consider performs *empirical risk minimization* (ERM). Its output is $A(S_n)$, i.e., a hypothesis h from a prespecified set $\mathcal{H}$ that minimizes the empirical risk over S_n based on a loss function $\mathcal{L} : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$. In statistical learning, performance is measured through this loss and the aim is to minimize the true risk $L_D(h) = \mathbb{E}_{z\sim D}\,\mathcal{L}(h, z)$. One can define classification problems, regression, and density estimation in such terms.

Before we formally introduce the concept of monotonicity, we mention related works that already report on non-monotone learning behavior. Duin [7] and Opper and Kinzel [8] describe the peaking phenomenon for classification: when the dimensionality is approximately equal to the size of the training set, the risk in terms of the zero-one loss and mean squared error has a maximum (it peaks). This happens for models that require estimates of the (pseudo-)inverse of the covariance matrix [9], such as linear regression.

Loog and Duin [10] describe what they call dipping: the evaluation risk attains a global minimum for some finite n. Even for $n \to \infty$ the risk never recovers. This phenomenon can occur when there is a mismatch between target (e.g. zero-one) and surrogate loss (e.g. hinge). Ben-David *et al.* [11] analyze this mismatch between surrogate and zero-one loss in more detail.

We focus on the setting where the loss the learner optimizes matches the loss it is evaluated with. Thus the observed behaviour in our examples cannot be explained through the dipping phenomenon. This makes our findings more unexpected and the open problem more appealing. Note, indeed, that our learner A (performing ERM) is implicitly associated with a specific loss $\mathcal{L}$ and set $\mathcal{H}$.

A.3. THE MONOTONICITY PROPERTY

The idea is that with an additional instance a learner should improve its performance in expectation over the training set. We need the following building block.

Definition 6 (local monotonicity)**.** *A learner A is locally or (D, n)-monotone with respect to a distribution D and an $n \in \mathbb{N}$ if*

$$\mathop{\mathbb{E}}_{S_{n+1}\sim D^{n+1}} L_D(A(S_{n+1})) \leq \mathop{\mathbb{E}}_{S_n\sim D^n} L_D(A(S_n)).$$

We may want to construct stronger properties from this, e.g. monotonicity for all n. Also, since the distribution D is unknown, we may want monotonicity to hold for any D on the domain $\mathcal{Z}$.

Definition 7 ($\mathcal{Z}$-monotonicity)**.** *A learner A is $\mathcal{Z}$-monotone if, for all $n \in \mathbb{N}$ and distributions D on $\mathcal{Z}$, it is (D, n)-monotone.*

A.4. EXAMPLES

We now turn to some illustrations and consider to what extent they are $\mathcal{Z}$-monotone. In the remainder, we refer to $\mathcal{Z}$-monotone as monotone. It will be clear from the context what $\mathcal{Z}$ is.

Example I: mean estimation of a normal distribution (monotone). We perform density estimation with a normal distribution with *fixed variance* $\sigma^2 > 0$ and *unknown mean*. The hypothesis class is $\mathscr{H}_\sigma = \left\{h : z \mapsto \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right) \mid \mu \in \mathbb{R}\right\}$. We choose the domain $\mathcal{Z} \subset [-1, 1]$. This choice ensures that any distribution D has a finite mean and finite variance. We use negative log-likelihood as loss. Thus ERM is equivalent to maximum likelihood (ML) estimation for this setting. The optimum that ERM finds is $\mu = \frac{1}{n}\sum_i z_i$. The expected risk equals

$$\mathop{\mathbb{E}}_{S_n\sim D} L_D(A(S)) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{\sigma_D^2}{2\sigma^2}\left(1 + \frac{1}{n}\right),$$

where σ_D^2 is the true variance of D. So the expected risk decreases monotonically in n.

Example II: variance estimation of a normal distribution (not monotone). We take the same domain and loss function as in Example I, but now estimate the variance, while keeping the mean fixed to 0. The hypothesis set becomes $\mathscr{H}_{\mu=0} = \left\{h : z \mapsto \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{z^2}{2\sigma^2}\right) \mid \sigma > 0\right\}$ and the ML estimate equals $\sigma = \frac{1}{n}\sum_i z_i^2$. This example does not obey the monotone principle. Consider a distribution D that only has support on $\{1, \frac{1}{10}\}$. Let D be given by the probability mass function $p(1) = \alpha$ and $p(\frac{1}{10}) = 1 - \alpha$. For $0 < \alpha < 0.0235$ one can easily check numerically that $\mathbb{E}_{S_1} L_D(A(S_1)) < \mathbb{E}_{S_2} L_D(A(S_2))$, showing that the monotonocity property does not hold.

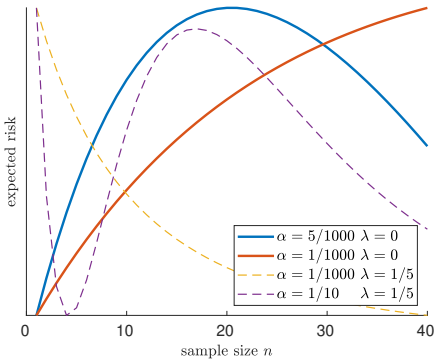


Figure A.1.: Non-monotone behavior as observed in Example III.

Example III: linear regression (not monotone). Take $\mathscr{H} = \{h \mapsto wx | w \in \mathbb{R}\}$ as hypothesis set and use the mean squared error as loss function. We choose the domain $\mathscr{Z} = \mathscr{X} \times \mathscr{Y}$, with $\mathscr{X} \subset [-1,1]$ and $\mathscr{Y} \subset [0,1]$. We define D through a probability mass function $p(x,y)$. Take $p(\frac{1}{10},1) = 1-\alpha$ and $p(1,1) = \alpha$, and $p(x,y) = 0$ otherwise. Again, one can find numerically that $\mathbb{E}_{S_1} L_D(A(S_1)) < \mathbb{E}_{S_2} L_D(A(S_2))$ for $0 < \alpha < 0.0047$. This shows this learner is not monotone.

Figure A.1 plots a rescaled version of the expected risk against the sample size n for several settings. The thick lines correspond to ERM. First of all, observe that by changing α, we can shift the peak. This shows that the behaviour is unrelated to the peaking behaviour [7], since peaking would occur at $n \approx d = 1$. Second, if we add λI to the empirical covariance matrix, which corresponds to L_2-regularization of w, we still observe non-monotone behavior, now even for larger values of α (see the dashed lines in Figure A.1).

Example IV: the memorize algorithm (monotone). Ben-David *et al.* [12] introduced this binary classifier. When evaluated on a test object x that is also present in the training set, this learner returns the label of that training object. In case multiple training examples share the same x, the majority vote is returned. In case the test object is not present in the training set, a default label is returned. This learner is monotone for any distribution under the zero-one loss as it only updates its decision on points that it observes.

A.5. RELATION TO LEARNABILITY

From learning theory we know that if the hypothesis class has finite VC-dimension (or other appropriate complexity), the excess risk of ERM is bounded. This bound will be tighter given a larger training set size n. PAC bounds hold with a particular probability, while we are concerned with the risk in expectation over the sample. However, even bounds that hold in expectation over the training sample will not rule out non-monotone behaviour. The expected risk can go up as long as the expected risk stays below the upper bound. Thus high probability or expected risk bounds are insufficient to guarantee monotonicity.

This is illustrated by our examples: Example VI is monotone but is not learnable [6].

Example III is learnable if a regularizer is added to the objective of ERM or if the hypothesis space $\mathscr{H}$ is restricted such that the norm of w is bounded. However, as we have seen in Figure A.1, we still can observe non-monotone behaviour in that case.

A.6. OPEN PROBLEM(S)

First and foremost, we are interested to identify, especially for commonly employed learners, on which domains $\mathscr{Z}$ they will or may not act monotonically. In view of the peaking behaviour, $\mathscr{Z}$-monotonicity for all n may be too strong for some settings. Perhaps monotonicity is only possible if n is larger than some N that may depend on $\mathscr{Z}$ and A. For Examples II and III it is an open problem whether they satisfy this weaker notion, and for which (smallest) N this notion is satisfied. Other related notions of monotonicity may also be of interest. For example, instead of demanding a lower loss, we may require that the loss does not degrade too much. Or we can demand the property to hold with high probability with respect to both samples.

More generally, we may ask: why and how does this behaviour occur? And maybe more importantly: how can we provably avoid non-monotone behaviour? What conditions does a learner need to satisfy to be monotone? Perhaps particular loss functions lead to monotone learners? What if we allow for learning under regularization or other strategies deviating from strict ERM, for example improper learners or randomized decision rules?

Perhaps it is always possible to find a D for a given $\mathscr{Z}$ on which learners are non-monotone. In that case, is it possible to avoid non-monotone behaviour under some assumptions on D? Realizeability or well-specification could be good candidate-assumptions on D. In fact, this raises the issue to what extent well-specified statistical models can actually be proven to behave monotonically. For instance, is Example II monotone if the problem is well-specified?

All in all, we believe the question of monotonicity of learning offers various tantalizing questions to study, some of which may yet have to be formulated.

A.7. BIBLIOGRAPHY

[1] T. Viering, A. Mey, and M. Loog, *Open problem: Monotonicity of learning,* in *Proceedings of the Thirty-Second Conference on Learning Theory*, Proceedings of Machine Learning Research, Vol. 99, edited by A. Beygelzimer and D. Hsu (Phoenix, USA, 2019) pp. 3198–3201.

[2] M. Loog, *Constrained parameter estimation for semi-supervised learning: the case of the nearest mean classifier,* in *ECML PKDD 2010* (Springer, 2010) pp. 291–304.

[3] W. M. Koolen, P. Grünwald, and T. van Erven, *Combining adversarial guarantees and stochastic fast rates in online learning,* in *NIPS* (2016) pp. 4457–4465.

[4] A. Liu, L. Reyzin, and B. D. Ziebart, *Shift-pessimistic Active Learning Using Robust Bias-aware Prediction,* in *Proceedings of AAAI-15* (2015) pp. 2764–2770.

[5] J. H. Krijthe and M. Loog, *Projected estimators for robust semi-supervised classification,* Machine Learning **106**, 993 (2017).

[6] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms* (Cambridge university press, 2014).

[7] R. P. W. Duin, *Small sample size generalization,* in *Proceedings of the Scandinavian Conference on Image Analysis*, Vol. 2 (1995) pp. 957–964.

[8] M. Opper and W. Kinzel, *Statistical mechanics of generalization,* in *Models of neural networks III* (Springer, 1996) pp. 151–209.

[9] S. Raudys and R. P. W. Duin, *Expected classification error of the Fisher linear classifier with pseudo-inverse covariance matrix,* Pattern recognition letters **19**, 385 (1998).

[10] M. Loog and R. P. W. Duin, *The dipping phenomenon,* in *Proceedings of the IAPR S+SSPR* (Springer, 2012) pp. 310–317.

[11] S. Ben-David, D. Loker, N. Srebro, and K. Sridharan, *Minimizing the misclassification error rate using a surrogate convex loss,* in *Proceedings ICML* (2012) pp. 83–90.

[12] S. Ben-David, N. Srebro, and R. Urner, *Universal learning vs. no free lunch results,* in *Philosophy and Machine Learning Workshop NIPS* (2011).

B

A Brief Prehistory of Double Descent

In their thought-provoking paper [1], Belkin et al. illustrate and discuss the shape of risk curves in the context of modern high-complexity learners. Given a fixed training sample size n, such curves show the risk of a learner as a function of some (approximate) measure of its complexity N. With N the number of features, these curves are also referred to as feature curves. A salient observation in [1] is that these curves can display, what they call, double descent: with increasing N, the risk initially decreases, attains a minimum, and then increases until N equals n, where the training data is fitted perfectly. Increasing N even further, the risk decreases a second and final time, creating a peak at $N = n$. This twofold descent may come as a surprise, but as opposed to what [1] reports, it has not been overlooked historically. Our letter draws attention to some original, earlier findings, of interest to contemporary machine learning.

Already in 1989, using artificial data, Vallet et al. [2] experimentally demonstrate double descent for *learning curves* of classifiers trained through minimum norm linear regression (MNLR, see [3])—termed the pseudo-inverse solution in [2]. In learning curves the risk is displayed as a function of n, as opposed to N for risk curves. What intuitively matters in learning behavior, however, is the sample size *relative* to the measure of complexity. This idea is made explicit in various physics papers on learning (e.g. [2, 4, 5]), where the risk is often plotted against $\alpha = \frac{n}{N}$. A first theoretical results on double descent, indeed using such α, is given by Opper et al. [4]. They proof that in particular settings, for N going to infinity, the pseudo-inverse solution improves as soon as one moves away from the peak at $\alpha = 1$.

Employing a pseudo-Fisher linear discriminant (PFLD, equivalent to MNLR), Duin [6] is the first to show feature curves on real-world data quite similar to the double-descent curves in [1]. Compare, for instance, Fig. 2 in [1] with Fig. 6 and 7 from [6]. Skurichina and Duin [7] demonstrate experimentally that increasing PFLD's complexity simply by adding random features can improve performance when $N = n$ (i.e., $\alpha = 1$). The benefit of some form of regularization has been shown already in [2]. For semi-supervised PFLD,

PNAS MS# 2020-01875 Letter to Editor. Accepted for electronic publications in the Proceedings of the National Academy of Sciences of the United States of America (https://www.pnas.org/).

Krijthe and Loog [8] demonstrate that unlabeled data can regularize, but also worsen the peak in the curve. Their work builds on the original analysis of double descent for the supervised PFLD by Raudys and Duin [9].

Interestingly, results from [4–7] suggest that some losses may not exhibit double descent in the first place. In [6, 7], the linear SVM shows regular monotonic behavior. Analytic results from [4, 5] show the same for the perceptron of optimal (or maximal) stability, which is closely related to the SVM [5].

The findings in [1] go, significantly, beyond those for the MNLR. Also shown, for instance, is double descent for 2-layer neural networks and random forests. Combining this with observations such as those from Loog et al. [10], which show striking multiple-descent learning curves (even in the underparameterized regime), the need to further our understanding of such rudimentary learning behavior is evident.

B.1. BIBLIOGRAPHY

[1] M. Belkin, D. Hsu, S. Ma, and S. Mandal, *Reconciling modern machine-learning practice and the classical bias–variance trade-off,* Proceedings of the National Academy of Sciences **116**, 15849 (2019).

[2] F. Vallet, J.-G. Cailton, and P. Refregier, *Linear and nonlinear extension of the pseudo-inverse solution for learning boolean functions,* Europhysics Letters **9**, 315 (1989).

[3] R. Penrose, *On best approximate solutions of linear matrix equations,* Mathematical Proceedings of the Cambridge Philosophical Society **52**, 17 (1956).

[4] M. Opper, W. Kinzel, J. Kleinz, and R. Nehl, *On the ability of the optimal perceptron to generalise,* Journal of Physics A: Mathematical and General **23**, L581 (1990).

[5] T. L. Watkin, A. Rau, and M. Biehl, *The statistical mechanics of learning a rule,* Reviews of Modern Physics **65**, 499 (1993).

[6] R. P. Duin, *Classifiers in almost empty spaces,* in *Proceedings of the 15th International Conference on Pattern Recognition*, Vol. 2 (IEEE, 2000) pp. 1–7.

[7] M. Skurichina and R. P. Duin, *Regularization by adding redundant features,* in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)* (Springer, 1998) pp. 564–572.

[8] J. H. Krijthe and M. Loog, *The peaking phenomenon in semi-supervised learning,* in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)* (Springer, 2016) pp. 299–309.

[9] S. Raudys and R. P. Duin, *Expected classification error of the Fisher linear classifier with pseudo-inverse covariance matrix,* Pattern Recognition Letters **19**, 385 (1998).

[10] M. Loog, T. Viering, and A. Mey, *Minimizers of the empirical risk and risk monotonicity,* in *Advances in Neural Information Processing Systems* (2019) pp. 7476–7485.

C

APPENDIX OF CHAPTER 3

C.1. BACKGROUND THEORY

C.1.1. MMD

The MMD quantity can be computed in practice by rewriting it as follows:

$$\begin{aligned}\mathrm{MMD}(\hat{P},\hat{Q}) &= \max_{\tilde{l}\in H_{\mathscr{L}}} \frac{1}{n_{\hat{P}}}\sum_{x\in\hat{P}}\langle\tilde{l},\psi_{K_{\mathscr{L}}}(x)\rangle_{K_{\mathscr{L}}} - \frac{1}{n_{\hat{Q}}}\sum_{x\in\hat{Q}}\langle\tilde{l},\psi_{K_{\mathscr{L}}}(x)\rangle_{K_{\mathscr{L}}} \\ &= \max_{\tilde{l}\in H_{\mathscr{L}}}\langle\tilde{l},\mu_{\hat{P}}-\mu_{\hat{Q}}\rangle_{K_{\mathscr{L}}} \qquad \text{(C.1)} \\ &= \Lambda_{\mathscr{L}}||\mu_{\hat{P}}-\mu_{\hat{Q}}||_{K_{\mathscr{L}}}. \qquad \text{(C.2)}\end{aligned}$$

In the first step we used that $\tilde{l}(x) = \langle\tilde{l},\psi_{K_{\mathscr{L}}}(x)\rangle_{K_{\mathscr{L}}}$ due to the reproducing property [1, p. 96]. Here $\psi_{K_{\mathscr{L}}}$ is the featuremap from $\mathscr{X}\to H_{\mathscr{L}}$. The second step follows from the linearity of the inner product. In (C.1) we defined $\mu_{\hat{P}} = \frac{1}{n_{\hat{P}}}\sum_{x\in\hat{P}}\psi_{K_{\mathscr{L}}}(x)$ and similarly for $\mu_{\hat{Q}}$, note that $\mu_{\hat{Q}},\mu_{\hat{P}}\in H_{\mathscr{L}}$. The last step follows from the fact that the vector in $H_{\mathscr{L}}$ maximizing the term in (C.1) is

$$\frac{\mu_{\hat{P}}-\mu_{\hat{Q}}}{||\mu_{\hat{P}}-\mu_{\hat{Q}}||_{K_{\mathscr{L}}}}\Lambda_{\mathscr{L}}$$

Because of the symmetry of $||\mu_{\hat{P}}-\mu_{\hat{Q}}||_{K_{\mathscr{L}}}$ with respect to $\hat{P}$ and $\hat{Q}$, this derivation also holds if we switch $\hat{P}$ and $\hat{Q}$. Therefore:

$$\max_{\tilde{l}\in H_{\mathscr{L}}}\left|\frac{1}{n_{\hat{P}}}\sum_{x\in\hat{P}}\tilde{l}(x) - \frac{1}{n_{\hat{Q}}}\sum_{x\in\hat{Q}}\tilde{l}(x)\right| = \Lambda_{\mathscr{L}}||\mu_{\hat{P}}-\mu_{\hat{Q}}||_{K_{\mathscr{L}}}$$

Therefore for all $\tilde{l}\in H_{\mathscr{L}}$ the following holds

$$\left|\frac{1}{n_{\hat{P}}}\sum_{x\in\hat{P}}\tilde{l}(x) - \frac{1}{n_{\hat{Q}}}\sum_{x\in\hat{Q}}\tilde{l}(x)\right| \le \mathrm{MMD}(\hat{P},\hat{Q}) \qquad \text{(C.3)}$$

We can compute the MMD quantity in practice by working out the norm with kernel products:

$$\begin{aligned}\mathrm{MMD}(\hat{P},\hat{Q}) &= \Lambda_{\mathscr{L}}\sqrt{\langle\mu_{\hat{Q}},\mu_{\hat{Q}}\rangle_{K_{\mathscr{L}}} - 2\langle\mu_{\hat{P}},\mu_{\hat{Q}}\rangle_{K_{\mathscr{L}}} + \langle\mu_{\hat{P}},\mu_{\hat{P}}\rangle_{K_{\mathscr{L}}}} \\ &= \Lambda_{\mathscr{L}}\sqrt{\mathrm{MMD}_{\mathrm{comp}}(\hat{Q},\hat{Q}) - 2\mathrm{MMD}_{\mathrm{comp}}(\hat{P},\hat{Q}) + \mathrm{MMD}_{\mathrm{comp}}(\hat{P},\hat{P})}\end{aligned}$$

where we introduced $\mathrm{MMD}_{\mathrm{comp}}(\hat{R},\hat{S}) = \frac{1}{n_{\hat{R}}n_{\hat{S}}}\sum_{x\in\hat{R},x'\in\hat{S}} K_{\mathscr{L}}(x,x')$.

C

C.1.2. DISCREPANCY

In this section we calculate the discrepancy analytically for the squared loss in the linear kernel as in [2]. We then extend the computation to any arbitrary kernel as in [3]. Finally, we prove the agnostic generalization bound in terms of the Discrepancy (Theorem 3). The theorems and proofs here were first given by Mansour *et al.* [2], Cortes and Mohri [3], and Cortes *et al.* [4] but we repeat them here for completeness.

Lemma 3 ([2])**.** *For $h, h' \in H$ we have*

$$\left|L_{\hat{P}}(h,h') - L_{\hat{Q}}(h,h')\right| = \left|\sum_{i=1}^{r}\bar{u}_i^2\lambda_i\right|. \tag{C.4}$$

Proof. We can show

$$L_{\hat{P}}(h,h') = \frac{1}{n_{\hat{P}}}(X_{\hat{P}}h - X_{\hat{P}}h')^T(X_{\hat{P}}h - X_{\hat{P}}h') = \frac{1}{n_{\hat{P}}}u^T X_{\hat{P}}^T X_{\hat{P}}u$$

using some algebra, where $u = h - h'$. Rewrite $L_{\hat{Q}}(h,h')$ similarly and subtract them to find

$$L_{\hat{P}}(h,h') - L_{\hat{Q}}(h,h') = u^T M u. \tag{C.5}$$

Since M is a real symmetric matrix, M is a normal matrix and admits an orthonormal eigendecomposition with real eigenvalues

$$M = \sum_{i}^{d} e_i\lambda_i e_i^T.$$

Here λ_i is the ith eigenvalue and e_i is the corresponding orthonormal eigenvector. Since M is normal its eigenvectors form an orthonormal basis for $\mathbb{R}^d$. Therefore we can express u in terms of e:

$$u = \sum_{i}^{d}\bar{u}_i e_i$$

Where $\bar{u}_i$ is the projection of u on e_i, $\bar{u}_i = e_i^T u$. Note $\bar{u}$ is a rotated version of u and therefore both have the same norm, $||u||_2 = ||\bar{u}||_2$. Now we can rewrite (C.5) as

$$u^T M u = \sum_{i}^{d} u^T e_i\lambda_i e_i^T u = \sum_{i=1}^{r}\bar{u}_i^2\lambda_i. \tag{C.6}$$

Note that M has $r = \mathrm{rank}(M)$ non-zero eigenvalues. Combining (C.5) and (C.6) and taking the absolute value on both sides shows the result. □

Now we are ready to compute the Discrepancy for the linear kernel.

Theorem 13 (Discrepancy computation [2])**.** *Assume K is the linear kernel, $K(x_i, x_j) = x_i^T x_j$, and l is the squared loss, then*

$$disc(\hat{P}, \hat{Q}) = 4\Lambda^2 \max_i |\lambda_i|.$$

where λ_i are the eigenvalues of $M_{\hat{P},\hat{Q}} = M$.

Proof. First we use Lemma 3.

$$\text{disc}(\hat{P}, \hat{Q}) = \max_{||\bar{u}|| \leq 2\Lambda} \left| \sum_i^r \bar{u}_i^2 \lambda_i \right| = \max \left(\max_{||\bar{u}|| \leq 2\Lambda} \sum_i^r \bar{u}_i^2 \lambda_i, \max_{||\bar{u}|| \leq 2\Lambda} \sum_i^r -\bar{u}_i^2 \lambda_i \right)$$

Now we solve the left term in the maximization. Observe that this is a weighted sum where each $\bar{u}_i$ weighs each eigenvalue λ_i. To maximize this quantity we put as much weight as possible on the largest postive eigenvalue: $u = e_{i_{\max}} 2\Lambda$, where $i_{\max} = \arg\max_i \lambda_i$. We find

$$\max_{||\bar{u}|| \leq 2\Lambda} \sum_i^d \bar{u}_i^2 \lambda_i = 4\Lambda^2 \max_i \lambda_i.$$

To solve the second maximization, introduce $\bar{\lambda}_i = -\lambda_i$. Then we maximize the same quantity as before but now λ replaced by $\bar{\lambda}$. It follows that the maximum is attained for $u = e_{i_{\min}} 2\Lambda$, where $i_{\min} = \arg\min_i \lambda_i$. We find

$$\text{disc}(\hat{P}, \hat{Q}) = 4\Lambda^2 \max\left(\lambda_i, \bar{\lambda}_i\right),$$

eliminating the maximum proves the result. □

Now we will describe how to compute the Discrepancy in case we work with an arbitrary kernel K. In this case we have to work in the RKHS $\mathscr{H}$ of the kernel K. Define $z(x) = \psi_K(x)$, and let $Z_{\hat{P}}$ be the datamatrix where each row is given by $z(x) : x \in \hat{P}$. Define $Z_{\hat{Q}}$ in the analogously. In this case Theorem 13 still holds, and the Discrepancy is given by the eigenvalues of M_Z:

$$M_Z = \frac{1}{n_{\hat{P}}} Z_{\hat{P}}^T Z_{\hat{P}} - \frac{1}{n_{\hat{Q}}} Z_{\hat{Q}}^T Z_{\hat{Q}} \tag{C.7}$$

However, now we run into problems, since for an arbitrary kernel K the dimensions of $\mathscr{H}$ can be very large or infinite, such as the case for the Gaussian kernel. Then we clearly cannot compute the matrix M_Z or its eigenvalues.

In the following we show that M_Z and M_K have the same eigenvalues. Then, to compute the Discrepancy with any kernel K, we can simply use the eigenvalues of M_K. First, let us define M_K.

$$M_K = K_{\hat{P}\hat{P}} D \tag{C.8}$$

where $K_{\hat{P}\hat{P}}$ is the $n_{\hat{P}} \times n_{\hat{P}}$ matrix where entry i, j is given by $K(x_i, x_j)$, and where D is a diagonal matrix where

$$D_{ii} = \begin{cases} \frac{1}{n_{\hat{P}}} - \frac{1}{n_{\hat{Q}}} & \text{if } x_i \in \hat{Q} \\ \frac{1}{n_{\hat{P}}} & \text{otherwise.} \end{cases}$$

Lemma 4 ([3])**.** *The eigenvalues of M_Z and M_K are the same.*

Proof. Recalling that $\hat{Q} \in \hat{P}$, and using some algebra, it can be shown that M_Z can be written as

$$M_Z = Z_{\hat{P}}^T D Z_{\hat{P}}.$$

Now we suggestively write M_Z and M_K as

$$M_Z = (Z_{\hat{P}}^T D) Z_{\hat{P}}$$
$$M_K = Z_{\hat{P}} (Z_{\hat{P}}^T D) = K_{\hat{P}\hat{P}} D$$

Here we used the fact that $K(x_i, x_j) = \langle \psi_K(x_i), \psi_K(x_j) \rangle_K$ (kernel trick) to rewrite M_K. Since the matrix product AB and BA have the same eigenvalues [3], M_K and M_Z have the same eigenvalues. □

Theorem 14. *Let K be any arbitrary PSD kernel. Then*

$$disc(\hat{P}, \hat{Q}) = 4\Lambda^2 \max_i |\lambda_i| = 4\Lambda^2 ||\lambda||_\infty \tag{C.9}$$

where λ is the vector of eigenvalues of the matrix M_K or M_Z, where M_K was defined in (C.8) *and M_Z was defined in* (C.7)*.*

Proof. First observe that Theorem 13 still holds, but we have to replace M by M_Z in case we use any arbitrary PSD kernel K. Then the result follows from Lemma 4. □

Finally, we give the proof of the generalization bound in terms of the Discrepancy for the agnostic setting. Note that this proof was already given by Cortes *et al.* [4], here we repeat their proof for completeness.

Proof of Theorem 3. Since $l(h(x), f(x)) \leq C$, we have that the squared loss l is μ-admissible [4] with $\mu = 2C$, meaning that

$$|l(h(x), f(x) - l(h'(x), f(x))| \leq 2C|h(x) - h'(x)| \tag{C.10}$$

holds for all $h, h' \in H$ and any $f : \mathscr{X} \to \mathscr{Y}$. Let $\tilde{f}$ be any arbitrary element from H. By adding and subtracting terms and applying the triangle inequality, we can show that

$$\begin{aligned} |L_{\hat{P}}(h, f) - L_{\hat{Q}}(h, f)| \leq & |L_{\hat{P}}(h, \tilde{f}) - L_{\hat{Q}}(h, \tilde{f})| + |L_{\hat{P}}(h, f) - L_{\hat{P}}(h, \tilde{f})| \\ & + |L_{\hat{Q}}(h, \tilde{f}) - L_{\hat{Q}}(h, f)|. \end{aligned}$$

The first term on the right hand side is by definition bounded by the Discrepancy. For the second term we can show

$$|L_{\hat{P}}(h, f) - L_{\hat{P}}(h, \tilde{f})| \leq 2C \frac{1}{n_{\hat{P}}} \sum_{x \in \hat{P}} |f(x) - \tilde{f}(x)| \leq 2C \max_{x \in \hat{P}} |f(x) - \tilde{f}(x)|.$$

The first inequality follows from applying (C.10) to each summand. We can bound the third term in the same way, since $\hat{Q} \in \hat{P}$. Bounding the first term using the Discrepancy and the last two terms with the bound above we find

$$|L_{\hat{P}}(h, f) - L_{\hat{Q}}(h, f)| \leq \text{disc}(\hat{P}, \hat{Q}) + 2\mu \max_{x \in \hat{P}} |f(x) - \tilde{f}(x)|$$

holds for all $\tilde{f} \in H$. The result follows from minimizing the right hand side with respect to $\tilde{f}$, bounding $L_{\hat{P}}(h,f) - L_{\hat{Q}}(h,f)$ with its absolute value and reordering terms. □

C.2. PROOFS

C.2.1. PROOF OF AGNOSTIC MMD WORST CASE BOUND (PROPOSITION 1)

Proof of Proposition 1. Let $\tilde{l}$ be any element from $H_{\mathscr{L}}$ and define $g_{\hat{P}} = \frac{1}{n_{\hat{P}}} \sum_{x \in \hat{P}} g(h,f)(x)$ and define $g_{\hat{Q}}$ similarly. Define $\tilde{l}_{\hat{P}} = \frac{1}{n_{\hat{P}}} \sum_{x \in \hat{P}} \tilde{l}(x)$ and $\tilde{l}_{\hat{Q}}$ analogously. Using the triangle inequality we can show

$$|L_{\hat{P}}(h,f) - L_{\hat{Q}}(h,f)| \leq |\tilde{l}_{\hat{P}} - \tilde{l}_{\hat{Q}}| + |g_{\hat{P}} - \tilde{l}_{\hat{P}}| + |g_{\hat{Q}} - \tilde{l}_{\hat{Q}}|.$$

The first term is bounded by the MMD, see (C.3). For the second term we have $|g_{\hat{P}} - \tilde{l}_{\hat{P}}| \leq \max_{x \in \hat{P}} |g(h,f)(x) - \tilde{l}(x)|$. This bound also holds also for the third term since $\hat{Q} \in \hat{P}$. Bounding the second and third term and maximizing over $h \in H$ we find that

$$|L_{\hat{P}}(h,f) - L_{\hat{Q}}(h,f)| \leq \mathrm{MMD}(\hat{P},\hat{Q}) + 2 \max_{h \in H, x \in \hat{P}} |g(h,f)(x) - \tilde{l}(x)|$$

holds for any $\tilde{l} \in H_{\mathscr{L}}$ and any $h \in H$. The result follows by choosing $\tilde{l}$ to minimize the right hand side, bounding $L_{\hat{P}}(h,f) - L_{\hat{Q}}(h,f)$ by the bound on its absolute value and reordering terms. □

C.2.2. ADJUSTING THE MMD TO THE LOSS AND HYPOTHESIS SET (THEOREM 2, COROLLARY 1)

In the main text we have given a sketch of the proof for the linear kernel. Here, we show a rigorous proof for the linear kernel, and afterward we give the proof for any arbitrary kernel K. The technique of the proof stays the same for any arbitrary kernel K, however, we have to do more bookkeeping.

Theorem 15 (Adjusted MMD linear kernel)**.** *Let l be the squared loss and assume $f \in H$ (realizable setting), furthermore assume K is the linear kernel, $K(x_i,x_j) = x_i^T x_j$. If $K_{\mathscr{L}}(x_i,x_j) = K(x_i,x_j)^2$ and $\Lambda_{\mathscr{L}} = 4\Lambda^2$, then $g(h,f) \in H_{\mathscr{L}}$ and thus $\eta_{MMD} = 0$.*

Proof of Theorem 15. Let $u = h - f$. Fix h and f, then we will write $g(x)$ as shorthand for $g(h,f)(x) = l(h(x),f(x))$. Then $g(x) = u(x)^2$. Since K is the linear kernel, we have that $h(x) = h^T x$, $f(x) = f^T x$ and $u(x) = u^T x$. Furthermore, we have $\mathscr{H} = \mathscr{X}$, and thus $\psi_K(x) = x$. Furthermore, $\psi_{K_{\mathscr{L}}} : \mathscr{H} \to \mathscr{H}_{\mathscr{L}}$ is given by [5, chap. 9.1][1]:

$$\psi_{K_{\mathscr{L}}}(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, x_3^2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3, x_4^2, \sqrt{2}x_1x_4, \sqrt{2}x_2x_4, \sqrt{2}x_3x_4, \ldots) \quad \text{(C.11)}$$

[1] Note that actually in [5] this kernel is defined as a polynomial kernel. In our case for this polynomial kernel we have that $R = 0$ and $d = 2$, resulting in the featuremap given in (C.11). This is often referred to as the squared kernel.

Since the featuremap of $K_{\mathscr{L}}$ exists, it is a PSD kernel, meaning that $K_{\mathscr{L}}(x, x') = \langle \psi_{K_{\mathscr{L}}}(x), \psi_{K_{\mathscr{L}}}(x') \rangle_{K_{\mathscr{L}}}$. Then we can write

$$g(x) = (u^T x)^2 = K(u, x)^2 = K_{\mathscr{L}}(u, x) = \langle \psi_{K_{\mathscr{L}}}(u), \psi_{K_{\mathscr{L}}}(x) \rangle_{K_{\mathscr{L}}}$$

thus $g \in \mathscr{H}_{\mathscr{L}}$ with $g = \psi_{K_{\mathscr{L}}}(u)$. Now what remains to show is that $g \in H_{\mathscr{L}}$, or in other words, that $||g||_{K_{\mathscr{L}}} \leq 4\Lambda^2$. We can show that

$$||g||_{K_{\mathscr{L}}} = \langle \psi(u), \psi(u) \rangle_{K_{\mathscr{L}}} = K_{\mathscr{L}}(u, u) = (u^T u)^2 = ||u||_K^2 \leq 4\Lambda^2$$

where the last step follows from that $h, f \in H$, and therefore $||u||_K \leq 2\Lambda$. This shows that $g(h, f) \in H_{\mathscr{L}}$, therefore $\eta_{\text{MMD}} = 0$. □

Before we prove the more general case for any kernel K, let us introduce some additional notation. Also, before we show the proof for the kernel $K_{\mathscr{L}}$, we first do the proof for the

Transformation		ψ_K		$\psi_{K'}$	
Space	$\mathscr{X}$	$\rightarrow$	$\mathscr{H}$	$\rightarrow$	$\mathscr{H}'$
Kernel			K		K'

Table C.1.: This table illustrates the notation used when 2 kernels are involved.

kernel K' which is slightly simpler, later we extend the result to $K_{\mathscr{L}}$. We define the squared kernel K' as:

$$K'(f, h) = \langle f, h \rangle_K^2 \tag{C.12}$$

Where $f \in \mathscr{H}$ and $g \in \mathscr{H}$, where $\mathscr{H}$ is the RKHS of K. We indicate $\mathscr{H}'$ as the RKHS of K'. We assume K is a PSD kernel. By definition of K' the kernel K' is a PSD kernel since a squared kernel of a PSD kernel is known to be PSD [1, Theorem 5.3]. Now we have two kernels we have two featuremaps: $\psi_K(x) : \mathscr{X} \rightarrow \mathscr{H}$ and $\psi_{K'} : \mathscr{H} \rightarrow \mathscr{H}'$. Note that the second featuremap can still be computed with (C.11). See Table C.1 for an overview of the notation used.

Recall that because K is PSD kernel we have that:

$$K(x, x') = \langle \psi_K(x), \psi_K(x') \rangle_K \tag{C.13}$$

For $x, x' \in \mathscr{X}$. Similarly for the kernel K' which is also PSD we have that:

$$K'(f, g) = \langle \psi_{K'}(f), \psi_{K'}(g) \rangle_{K'} \tag{C.14}$$

For $f, g \in \mathscr{H}$. Again we define u as:

$$u = h - f$$

Theorem 16 (Adjusted MMD for K')**.** *Let l be the squared loss and assume $f \in H$ (realizable setting), and assume K is a PSD kernel. If $K'(f, h) = \langle f, h \rangle_K^2$ and $\Lambda_{\mathscr{L}} = 4\Lambda^2$, then $g(h, f) \in H' = \{h \in \mathscr{H}' : ||h||_{K'} \leq \Lambda_{\mathscr{L}}\}$ where $\mathscr{H}'$ is the RKHS of K'.*

Proof. We have $g(x) = u(x)^2$. Since $h, f \in \mathscr{H}$, $u \in \mathscr{H}$ and thus

$$u(x) = \langle u, \psi_K(x) \rangle_K$$

The first step is to show that the function $g \in \mathscr{H}'$. By definition:

$$g(x) = u(x)^2 = \langle u, \psi_K(x) \rangle_K^2$$

Now we can easily recognize our definition of K' in this equation (compare with (C.12)), thus we note that:

$$g(x) = K'(u, \psi_K(x)) = \langle \psi_{K'}(u), \psi_{K'}(\psi_K(x)) \rangle_{K'}$$

Where the second equality is obtained by applying (C.14). We observe that g corresponds to the vector $\psi_{K'}(u) \in \mathscr{H}'$, and thus we have that $g \in \mathscr{H}'$.

The second step is to show that $g \in H'$, in other words that $||g||_{K'} \leq 4\Lambda^2$. Since $g = \psi_{K'}(u) \in \mathscr{H}'$ the norm of g in K' is given by

$$||g||_{K'}^2 = \langle \psi_{K'}(u), \psi_{K'}(u) \rangle_{K'}.$$

Now we can use (C.14) to rewrite this in terms of K'. We obtain:

$$||g||_{K'}^2 = K'(u, u) \tag{C.15}$$

Using the definition of K' we find:

$$K'(u, u) = \langle u, u \rangle_K^2 = ||u||_K^4 \tag{C.16}$$

Now since $||h||_K \leq \Lambda$ and $||f||_K \leq \Lambda$ since $h, f \in H$, we have

$$||u||_K = ||h - f||_K \leq 2\Lambda$$

Combining this with Equations C.15 and C.16 we find that:

$$||g||_{K'} = ||u||_K^2 \leq 4\Lambda^2$$

Thus we have shown that $g \in \mathscr{H}'$. □

However, do we now have $g \in H_{\mathscr{L}}$? As of now we defined the kernel $K'(f, h)$ so that it operates on $f, g \in \mathscr{H}$. This does not coincide with the kernel $K_{\mathscr{L}}$. Therefore, we will now argue that $\mathscr{H}' = \mathscr{H}_{\mathscr{L}}$, and thus that in general the result generalizes to any kernel K, proving Theorem 2.

Proof of Theorem 2. By definition of $K_{\mathscr{L}}$ we have that:

$$K_{\mathscr{L}}(x, x') = K(x, x')^2$$

Now using (C.13) we can show that:

$$K_{\mathscr{L}}(x, x') = K(x, x')^2 = \langle \psi_K(x), \psi_K(x') \rangle_K^2$$

Observe that this coincides with the definition of K' ((C.12)), thus we can write this as:

$$K_{\mathscr{L}}(x,x') = \langle \psi_K(x), \psi_K(x') \rangle_K^2 = K'(\psi_K(x), \psi_K(x'))$$

Now using (C.14) we can write this as:

$$K_{\mathscr{L}}(x,x') = K'(\psi_K(x), \psi_K(x')) = \langle \psi_{K'}(\psi_K(x)), \psi_{K'}(\psi_K(x')) \rangle_{K'}$$

in other words, we see that the kernel product of $K_{\mathscr{L}}$ can be computed in the RKHS of the kernel K'. Thus, the RKHS of K' and $K_{\mathscr{L}}$ coincide! Thus we have that $\mathscr{H}' = \mathscr{H}_{\mathscr{L}}$. Therefore, Theorem 16 implies that we can generalize all results in terms of K' to the kernel $K_{\mathscr{L}}$. Therefore, g is also in the RKHS of $K_{\mathscr{L}}$, and in particular we have that $g \in H_{\mathscr{L}}$, and therefore $\eta_{\text{MMD}} = 0$. □

Remark 3. *Another way to understand this is to see that the featuremap of $K_{\mathscr{L}}$ is given by $\psi_{K_{\mathscr{L}}}(x) = \psi_{K'}(\psi_K(x))$ and thus maps to the space $\mathscr{H}'$, and from this it follows that $\mathscr{H}' = \mathscr{H}_{\mathscr{L}}$.*

Proof of Corollary 1. Theorem 2 tells us to choose $K_{\mathscr{L}}(x,x') = K(x,x')^2$ to obtain $\eta_{\text{MMD}} = 0$. We can show

$$K'(x,x') = K(x,x')^2 = \exp\left(-\frac{2||x-x'||^2}{2\sigma^2}\right) = \exp\left(-\frac{||x-x'||^2}{2\sigma'^2}\right),$$

where we absorbed the factor of 2 in the exponent in $\sigma_{\mathscr{L}}$, so $\sigma_{\mathscr{L}} = \frac{\sigma}{\sqrt{2}}$. □

C.2.3. MMD COMPUTATION (THEOREM 5 AND COROLLARY 2)

First we prove the Theorem 5 in case K is the lineair kernel for $d = 2$, afterward we extend the proof to any dimension, and finally we prove Theorem 5 for any PSD kernel.

Theorem 17 (MMD Computation linear kernel $d = 2$)**.** *Let $K_{\mathscr{L}}(x_i,x_j) = K(x_i,x_j)^2$ and $\Lambda_{\mathscr{L}} = 4\Lambda^2$. Furthermore, assume K is the linear kernel, $K(x_i,x_j) = x_i^T x_j$ and $d = 2$, then*

$$MMD(\hat{P},\hat{Q}) = 4\Lambda^2 ||\lambda||_2. \tag{C.17}$$

Proof. If K is the linear kernel, $\mathscr{H} = \mathscr{X}$ and $K_{\mathscr{L}}$ defines a featuremap $\psi_{K_{\mathscr{L}}}(x) : \mathscr{X} \to \mathscr{H}_{\mathscr{L}}$ which is given by $\psi_{K_{\mathscr{L}}}(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$, see [5, chap. 9.1]. From (C.2) we find

$$\text{MMD}(\hat{P},\hat{Q}) = \Lambda_{\mathscr{L}} ||\mu_{\hat{P}} - \mu_{\hat{Q}}||_{K_{\mathscr{L}}}$$

where $\mu_{\hat{P}} = \frac{1}{n_{\hat{P}}} \sum_{x \in \hat{P}} \psi_{K_{\mathscr{L}}}(x)$ and $\mu_{\hat{Q}}$ is analogously defined. Now using the fact $\mu_{\hat{P}} - \mu_{\hat{Q}} = (M_{11}, M_{22}, \sqrt{2}M_{12})^T$ and some algebra we can show

$$\text{MMD}(\hat{P},\hat{Q}) = 4\Lambda^2 ||M||_F = 4\Lambda^2 ||\lambda||_2$$

where the second equality follows since M is a real symmetric matrix and thus its eigenvalues are equal to its singular values (up to a sign change). □

Let us first generalize Theorem 17 to any arbitrary dimension d, afterward we extend the result to any kernel K.

Theorem 18 (MMD Computation linear kernel arbitrary dimension)**.** *If $K_{\mathscr{L}}(x_i, x_j) = K(x_i, x_j)^2$ and $\Lambda_{\mathscr{L}} = 4\Lambda^2$. Furthermore, assume K is the linear kernel, $K(x_i, x_j) = x_i^T x_j$ and the dimension of the input space $\mathscr{X}$ is d, then*

$$MMD(\hat{P}, \hat{Q}) = 4\Lambda^2 ||\lambda||_2.$$

Proof. We can show using (C.11) that the vector $\mu_{\hat{P}} - \mu_{\hat{Q}}$ becomes

$$(\mu_{\hat{P}} - \mu_{\hat{Q}})^T = (M_{11}, M_{22}, \sqrt{2}M_{12}, M_{33}, \sqrt{2}M_{13}, \sqrt{2}M_{23}, M_{44}, \sqrt{2}M_{14}, \sqrt{2}M_{24}, \sqrt{2}M_{34}, \ldots)^T.$$

Observe that the entry M_{ii} appears only once in $\mu_{\hat{P}} - \mu_{\hat{Q}}$, and any entry M_{ij} where $i \neq j$, appears as $\sqrt{2}M_{ij}$. Furthermore, note that the diagonal M_{ii} only occurs once in the matrix M. However, any element M_{ij} appears twice in M (since M is symmetric). Therefore

$$(\mu_{\hat{P}} - \mu_{\hat{Q}})^T (\mu_{\hat{P}} - \mu_{\hat{Q}}) = \sum_i M_{ii}^2 + 2 \sum_{i \neq j} M_{ij}^2,$$

is a sum of all entries (squared) of M, and therefore

$$||\mu_{\hat{P}} - \mu_{\hat{Q}}||_{K_{\mathscr{L}}} = ||M||_F,$$

as before. The rest of the proof is identical to the proof of Theorem 17 ($d = 2$). □

Note that due to our careful ordering of the featuremap of the squared kernel, given in (C.11), we have that this featuremap is still properly defined even if the dimension of $d \to \infty$, such as for a Gaussian kernel. Now we are ready to prove Theorem 5.

Proof of Theorem 5. To show the result holds for any arbitrary kernel K, we have to work in the RKHS of K, thus everywhere x needs to be replaced by $z(x)$ as in Appendix C.1.2, and then we replace M with M_Z. Then Theorem 18 still holds, since the featuremap $\psi(x)_{K'}$ is still given by the featuremap of the squared kernel, (C.11), however in this case the featuremap is with respect z instead of x. This does not influence the proof. We showed that this holds for any dimension d, and the featuremap still exists if $d \to \infty$, thus our results hold for a kernel K with arbitrary dimension of the RKHS $\mathscr{H}$.

For the Gaussian kernel we cannot compute the matrix M. Instead, since M and M_K have the same eigenvalues, see Lemma 4, we can compute the MMD instead using the eigenvalues of M_K, as in Appendix C.1.2. □

Remark 4. *Note that*

$$MMD(\hat{P}, \hat{Q}) \neq 4\Lambda^2 ||M_K||_F.$$

These are not equal, since the matrix M_K is not symmetric. Therefore, the eigenvalues of M_K are not the same as the singular values of M_K (as was the case for M, which is symmetric).

Proof of Corollary 2. Comparing Equation C.9 and Equation 3.6 and noting $||\lambda||_\infty \leq ||\lambda||_2$ shows the result. □

C.2.4. PROBABILISTIC ANALYSIS (LEMMA 1, PROPOSITION 2 AND THEOREM 6)

Proof of Lemma 1. We can show that:

$$\left|L_{\hat{P}}(h,h') - L_{\hat{Q}}(h,h')\right| = \left|\sum_{i=1}^{r} \bar{u}_i^2 \lambda_i\right| \leq \sum_{i=1}^{r} \bar{u}_i^2 |\lambda_i|$$

where the equality follows from Equation C.4 and the inequality follows from the triangle inequality. Next, bound $L_{\hat{P}}(h,h') - L_{\hat{Q}}(h,h')$ using the bound above and reorder terms. The result is obtained after applying the expectation w.r.t. u on both sides and applying the linearity of the expectation. □

Proof of Proposition 2. Computing $G(2\Lambda e_1, M)$ which will be found to be exactly equal to the Discrepancy. After combining this fact with Lemma 1 will result in the desired equality, the inequality follows from Corollary 2. □

Proof of Theorem 6. We can show that

$$\mathop{\mathbb{E}}_{u} G(u,M) = \frac{1}{\sqrt{r}} \mathrm{MMD}(\hat{P},\hat{Q}) \leq \mathrm{disc}(\hat{P},\hat{Q}). \tag{C.18}$$

This equality can be shown by working out the expectation and canceling terms and recognizing the definition of the MMD from (3.6). The inequality follows from $||\lambda||_2 \leq \sqrt{r}||\lambda||_\infty$. The final result follows by combining Equation C.18 with Lemma 1. □

C.2.5. PROOF OF NUCLEAR DISCREPANCY BOUND (THEOREM 7)

Before we can show the proof of the Nuclear Discrepancy bound, we need the following lemma:

Lemma 5. *Let $p(u)$ be uniform over all $u \in U$. Then*

$$\mathop{\mathbb{E}}_{u} \bar{u}_1^2 = \frac{4\Lambda^2}{r+2}.$$

Proof. By comparing the volume of a sphere of radius 2Λ and the volume of a sphere of radius $w = ||u||_2$, we can show that for this distribution $p(u)$ we have that

$$p(w) = \frac{w^{(r-1)} r}{(2\Lambda)^r}.$$

Then it is straightforward to show that

$$\mathop{\mathbb{E}}_{u} ||u||_2^2 = \frac{r}{r+2} 4\Lambda^2.$$

by integration of $p(w)$. From the symmetry of $p(u)$ it follows that $\mathbb{E}_u \bar{u}_1^2 = \mathbb{E}_u \bar{u}_i^2$ for all i. From this fact and the linearity of the expectation the result follows. □

Proof of Theorem 7. We can show that

$$\mathop{\mathbb{E}}_{u} G(u, M) = \sum_i |\lambda_i| \mathop{\mathbb{E}}_{u} \bar{u}_i^2 = ||\lambda||_1 \mathop{\mathbb{E}}_{u} \bar{u}_1^2 = \frac{4\Lambda^2}{r+2} ||\lambda||_1.$$

The first equality follows from switching expectation and sum. The second equality follows from symmetry of $p(u)$. The last equality follows from Lemma 5. The bound can be obtained by combining with Theorem 8. The inequalities follow from the vector norm inequalities $||\lambda||_1 \leq \sqrt{r}||\lambda||_2 \leq r||\lambda||_\infty$. □

C.3. REMARK ON PROBABILISTIC ANALYSIS AND CHOICE OF U_s

The remark in this section will explain why instead of U, we need to take U_s (to be defined below). The problem stems from the fact that if we choose $p(u)$ uniform on U, it may seem unclear what it means for u to be randomly sampled from an infinite dimensional sphere uniformly.

We will use the notation of Appendix C.1.2, since we will work with a kernel K with a high-dimensional $\mathscr{H}$, in order to highlight the problem that a lot of eigenvalues may be zero. We are analyzing what happens to

$$\mathop{\mathbb{E}}_{u} G(u, M_Z) \tag{C.19}$$

for arbitrary distributions, for example, for the uniform distribution $p(u)$, in case the length of a vector $z(x)$ is infinite such as with a Gaussian kernel (then M has infinite eigenvalues, but only r are non-zero).

The RKHS of K, $\mathscr{H}$, is then of infinite dimension. We split $\mathscr{H}$ in two parts: $\mathscr{H}_s = span(Z_{\hat{P}})$, and its orthogonal complement $\mathscr{H}_s^\perp$. Then for any vector in $a \in \mathscr{H}$, $a = a_s + a_s^\perp$, where $a_s \in \mathscr{H}_s$ and $a_s^\perp \in \mathscr{H}_s^\perp$. In particular we have that after training a kernel regularized model, we have $h \in a_s$ due to the regularization term in the training procedure. Furthermore, for any observed f, we have

$$L_{\hat{P}}(h, f) = L_{\hat{P}}(h, f_s), \tag{C.20}$$

since $Z_{\hat{P}} f_s^\perp = 0$. The same thing holds for $L_{\hat{Q}}(h, f)$. Therefore, we may consider it redundant to consider f, h, and we may limit our analysis to f_s and h_s. In addition we have that

$$G(u, M_Z) = G(u_s, M_Z), \tag{C.21}$$

since u is projected on eigenvectors of M_Z, and only eigenvectors in $span(Z_{\hat{P}})$ have non-zero eigenvalue, and thus $u_s^\perp$ only has components that correspond to eigenvalues that are zero. Thus any u has the same objective as the corresponding u_s. Therefore, instead of defining a pdf over U, we define a pdf over $U_s = \{u \in \mathscr{H}_s : ||u||_K \leq 2\Lambda\}$. Then by construction the dimension of u is at most $r = \text{rank}(M_Z) \leq n_{\hat{P}}$, which is always finite. Then sampling u is a well defined procedure even in infinite dimensional RKHS.

C.4. COMPUTATION OF THE DECOMPOSITION OF THE PROBABILISTIC BOUNDS

To compute each term of $G(u, M_Z)$, we can compute the eigendecomposition of M_K to compute the eigenvalues, however we also need to know $\bar{u}_i$ for each i. The computation of $\bar{u}_i$, the projection of u onto the eigenvector v_i of M_Z is non-trivial to compute in case kernels are used. Observe that here v_i is the ith eigenvector and not a component. Here we assume v_i is not normalized to unit norm (which is why we write it differently from e_i). We give a detailed description in this appendix how to compute $\bar{u}_i$. In this case the equation for $\bar{u}_i$ is:

$$\bar{u}_i = \frac{u^T v_i}{\sqrt{v_i^T v_i}} \tag{C.22}$$

The difficulty in this derivation is finding the vector v_i in case kernels are used. Then we need to find v_i expressed in terms of the datamatrix Z. Then we can apply the 'kernel trick' to compute (C.22).

By the eigenvalue equation of M_Z we have:

$$M_Z v_i = \lambda_i v_i \tag{C.23}$$

In case of the linear kernel it is straightforward to compute v_i. However, to compute v_i for any K, we have to take extra steps. First we show that v_i can be expressed in terms of the datamatrix Z, and afterward we find this expression of v_i in terms of Z. Note that:

$$M_Z v_i = \sum_{j=1}^{n_{\hat{P}}} d_j z_j z_j^T v_i = \sum_{j=1}^{n_{\hat{P}}} (z_j^T v_i) d_j z_j = \lambda_i v_i$$

Thus we have that:

$$\sum_{j=1}^{n_{\hat{P}}} \frac{(z_j^T v_i) d_j}{\lambda_i} z_j = v_i$$

Thus we have that each eigenvector v_i is a linear combination of the vectors z_j. Here the sum is taken over all objects $z(x) : x \in \hat{P}$. Since $\hat{Q} \subseteq \hat{P}$, this includes all data the active learner has access to. Then we can write each eigenvector v_i as:

$$v_i = Z_{\hat{P}}^T \alpha_i \tag{C.24}$$

Thus we can express each vector v_i using the datamatrix $Z_{\hat{P}}$. Now we will have to find the vector α_i to find v_i. We substitute the equation above in (C.23) to obtain:

$$M_Z Z_{\hat{P}}^T \alpha_i = \lambda_i Z_{\hat{P}}^T \alpha_i$$

Now we multiply left with $D Z_{\hat{P}}$ on both sides to obtain:

$$D Z_{\hat{P}} M_Z Z_{\hat{P}}^T \alpha_i = \lambda_i D Z_{\hat{P}} Z_{\hat{P}}^T \alpha_i$$

Observe that this is equal to:

$$M_K^T M_K^T \alpha_i = \lambda_i M_K^T \alpha_i$$

Where M_K was defined in (C.8). Now we define $\beta_i = M_K^T \alpha_i$. Then we find:

$$M_K^T \beta_i = \lambda_i \beta_i \tag{C.25}$$

We can compute the eigenvectors β by computing the eigendecomposition of M_K^T. This is possible even when using kernels, since M_K is expressed in terms of the kernel matrix. However we require the vector α_i to compute the eigenvector v_i. Thus now we will aim to express α_i in terms of β_i. Observe that if we multiply (C.25) by $(M_K^T)^{-1}$ on both sides we obtain:

$$\beta_i = \lambda_i (M_K^T)^{-1} \beta_i \tag{C.26}$$

Now observe that due to the definition of β_i we have that:

$$\beta_i (M_K^T)^{-1} = \alpha_i \tag{C.27}$$

Combining (C.26) and (C.27) we find that:

$$\alpha_i = \frac{\beta_i}{\lambda_i}$$

Substituting this in (C.24) we find the vector v_i:

$$v_i = Z_{\hat{P}}^T \frac{\beta_i}{\lambda_i} \tag{C.28}$$

Now we have found v_i. Now we can proceed to compute u_i.

Note that due to the representer theorem we have that the hyperplane of each model can be written as a linear combination of the data:

$$u = f - h = Z_{\hat{D}}^T c' - Z_{\hat{Q}}^T c \equiv Z_{\hat{D}}^T \tilde{c} \tag{C.29}$$

Here f is given as a linear combination of $Z_{\hat{D}}$, which we define as the complete datamatrix. This datamatrix includes the training and test set, since f in our experiments was obtained by training on the whole dataset where the original binary labels of the dataset are used (in the realizeable setting). However note that for any $f \in H$ the model f can be written in this way. Similarly, since h is trained on the dataset $\hat{Q}$, we can write h as a linear combination of objects $Z_{\hat{Q}}$. Combining (C.28) and (C.29) with (C.22) we find that:

$$\bar{u}_i = \frac{\tilde{c} Z_{\hat{D}} Z_{\hat{P}}^T \frac{\beta_i}{\lambda_i}}{\sqrt{\frac{\beta_i^T}{\lambda_i} Z_{\hat{P}} Z_{\hat{P}}^T \frac{\beta_i}{\lambda_i}}} = \frac{\tilde{c} K_{\hat{D}\hat{P}} \beta_i}{\sqrt{\beta_i K_{\hat{P}\hat{P}} \beta_i}}$$

C.5. EXPERIMENTAL SETTINGS AND DATASET CHARACTERISTICS

The active learning methods are evaluated on the datasets shown in Table C.2. The datasets marked with * were provided by Cawley and Talbot [6]. Other datasets originate from the UCI Machine Learning repository [7], except the MNIST dataset [8] which is a standalone dataset.

Table C.2.: Characteristics of evaluation datasets.

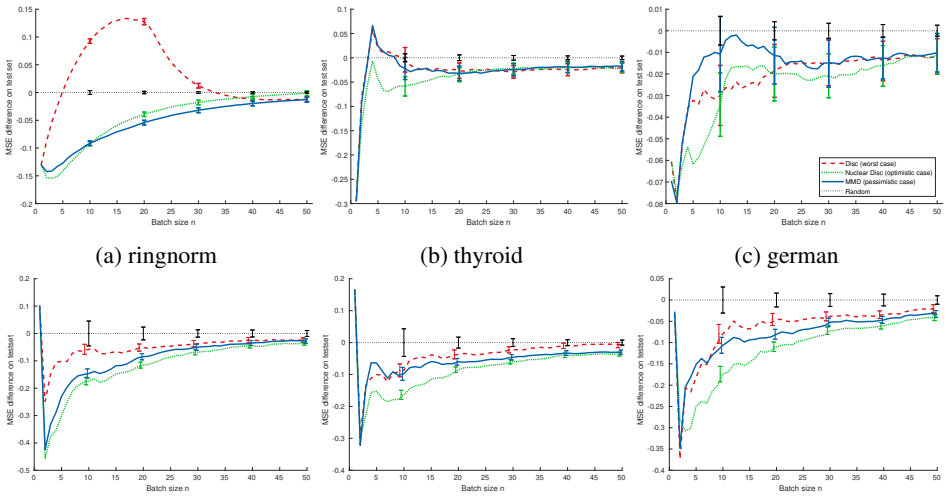
Dataset	# Objects	# Positive class	Dimensionality
vehicles	435	218	18
heart	297	137	13
sonar	208	97	60
thyroid*	215	65	5
ringnorm*	1000	503	20
ionosphere	351	126	33
diabetes	768	500	8
twonorm*	1000	500	20
banana*	1000	439	2
german	1000	700	20
splice	1000	541	60
breast	699	458	9
mnist 3vs5	1000	484	784
mnist 7vs9	1000	510	784
mnist 5vs8	1000	535	784

The parameter settings used are displayed in Table C.3. To obtain these hyperparameters we repeated the following procedure multiple times. We randomly select 25 examples from the dataset and label these. We train a KRLS model on these samples and evaluate the MSE on all unselected objects. The hyperparameters that result in the best performance after averaging are used in the active learning experiments.

Table C.3.: Table with parameters used for the benchmark datasets

Dataset	σ	$\log_{10}(\mu)$
vehicles	5.270	-3.0
heart	5.906	-1.8
sonar	7.084	-2.6
thyroid	1.720	-2.6
ringnorm	1.778	-3.0
ionosphere	4.655	-2.2
diabetes	2.955	-1.4
twonorm	5.299	-2.2
banana	0.645	-2.2
german	4.217	-1.4
splice	9.481	-2.6
breast	4.217	-1.8
mnist 3vs5	44.215	-6.0
mnist 7vs9	44.215	-3.6
mnist 5vs8	44.215	-8.9

C.6. RESULTS OF THE AGNOSTIC SETTING



(a) ringnorm (b) thyroid (c) german

(d) mnist 3vs5 (e) mnist 7vs9 (f) mnist 5vs8

Figure C.1.: Learning curves for several datasets for the agnostic setting. Results are averaged over 100 runs. Observe that compared to the realizable setting the variability of the performance increases and therefore performance differences become less significant. Due to unexpected effects of $\eta > 0$ the ranking of the methods may change.

For completeness we discuss the results of the agnostic setting where the original binary

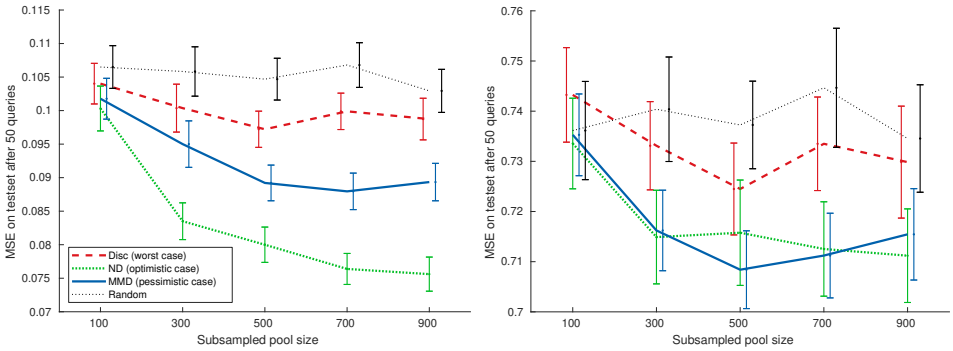
labels are used. In this setting $\eta \neq 0$, but η will be small due to our choice of hyperparameters, and therefore we ignore it during active learning (since we also cannot estimate it unless we have the labels of $\hat{P}$). Several illustrating learning curves are shown in Figure C.1, all results are summarized in Table C.4, all learning curves can be found in Appendix C.8.3.

The curves are less smooth and have larger standard errors compared to the realizable setting. Therefore the active learning methods are harder to distinguish which is reflected in Table C.4 by larger standard deviations and more bold numbers in a single row. Observe that the ranking of the methods can also change, see for example the learning curve on ringnorm: in the realizable setting the ND improved upon the Discrepancy, while in the agnostic setting the reverse is the case for large budgets. In this setting, sometimes the Discrepancy performs the best. From Table C.4 we can see that the trends observed in the realizable setting are still observed in the agnostic setting: the ND improves more upon the MMD than the reverse, however, the trend is weaker. This is likely the case because for this setting η_{MMD} and η_{disc} are non-zero, and therefore our theoretical analysis is weakened. Finally, observe that for the MNIST dataset, the learning curves and results as summarized by the AULC for 5vs8 are almost completely identical as in the realizeable setting. Similarly, for 3vs5 differences are also quite small. This indicates that MNIST is very close to realizeable with these found hyperparameter settings.

Table C.4.: Area Under the mean squared error Learning Curve (AULC) for the strategies in the agnostic setting, averaged over 100 runs. Bold indicates the best result, or results that are not significantly worse than the best result, according to a paired t-test ($p = 0.05$). Parenthesis indicate standard deviation.

Dataset	Random	Discrepancy	MMD	Nuclear Discrepancy
vehicles	25.8 (4.7)	**22.9 (2.7)**	23.9 (3.2)	23.5 (2.6)
heart	34.9 (4.0)	**32.7 (3.3)**	**32.4 (3.8)**	**32.4 (3.7)**
sonar	40.6 (4.3)	39.8 (4.4)	38.3 (3.6)	**37.3 (4.2)**
thyroid	17.9 (3.4)	16.4 (3.5)	16.3 (3.1)	**15.7 (2.9)**
ringnorm	35.9 (1.4)	37.5 (0.7)	**33.1 (1.0)**	33.5 (1.0)
ionosphere	28.9 (3.4)	26.7 (2.5)	**26.3 (2.7)**	27.3 (3.4)
diabetes	40.5 (3.2)	**39.6 (3.2)**	**39.7 (3.0)**	40.2 (2.7)
twonorm	19.3 (2.4)	17.3 (1.6)	17.0 (1.6)	**16.2 (1.3)**
banana	32.1 (3.4)	**28.5 (3.4)**	28.6 (2.9)	**27.8 (2.5)**
german	42.2 (3.2)	**40.8 (2.3)**	41.1 (2.6)	**40.6 (2.4)**
splice	45.4 (3.1)	45.2 (3.5)	44.6 (2.8)	**43.7 (2.6)**
breast	11.7 (2.7)	**10.3 (1.7)**	**10.1 (1.7)**	**10.1 (1.8)**
mnist 3vs5	30.6 (4.5)	28.1 (2.5)	26.1 (2.2)	**25.0 (1.8)**
mnist 7vs9	27.5 (3.6)	25.5 (2.4)	24.6 (2.0)	**23.2 (1.6)**
mnist 5vs8	30.2 (3.4)	26.9 (2.7)	26.1 (2.3)	**24.5 (2.1)**

C.7. INFLUENCE OF SUBSAMPLING ON PERFORMANCE.



(a) splice realizeable (b) splice agnostic

Figure C.2.: Dataset size versus performance of the active learners on splice. We observe that for larger dataset sizes, the active learners typically improve with respect to random sampling, but the improvement levels off for large dataset sizes.

We perform an additional experiment on the splice dataset to see how subsampling affects performance. To this end we measure the performance while we vary the pool size by changing the amount of subsampling, The subsampled pool is used as training set (this is the pool from which active learners can select queries, $\hat{P}$), all remaining samples are used as testset $\hat{T}$. Furthermore we use the same experimental protocol as for the other experiments.

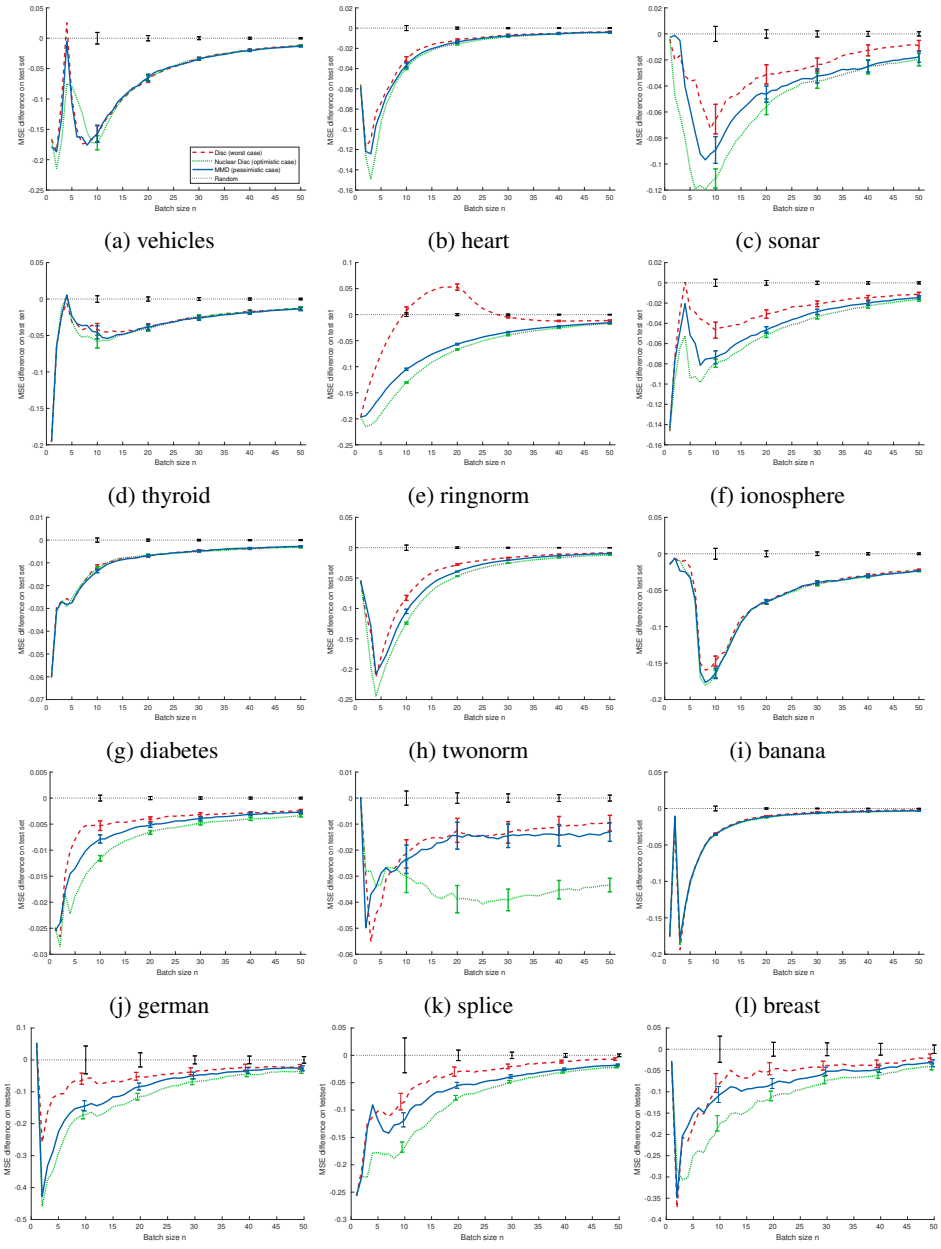
We display the performance of the active learners in terms of MSE on the testset after 50 queries in Figure C.2 for both the realizeable and agnostic setting. The curve is averaged over 100 runs. Error bars represent the 95% confidence interval computed using the standard error. As expected, the trends in the realizeable setting are more clear, while due to model misspecification and other effects in the agnostic case performance differences are less clear due to larger standard deviations.

For small pool sizes all active learners experience a drop in performance. In this case the probability is large that 'good' queries may be missing because of an unlucky draw from the dataset. For larger pool sizes most active learners perform better. For larger pool sizes the performance at some point levels. At this point the pool contains sufficient representative samples for it to contain all possible 'good' queries the active learners will be looking for. The experiment provides evidence that if finer subsampling is used, methods typically improve in performance up to a point where performance levels off.

C

C.8. ADDITIONAL EXPERIMENTAL RESULTS

C.8.1. LEARNING CURVES ON ALL DATASETS FOR THE REALIZABLE SETTING



(a) vehicles (b) heart (c) sonar

(d) thyroid (e) ringnorm (f) ionosphere

(g) diabetes (h) twonorm (i) banana

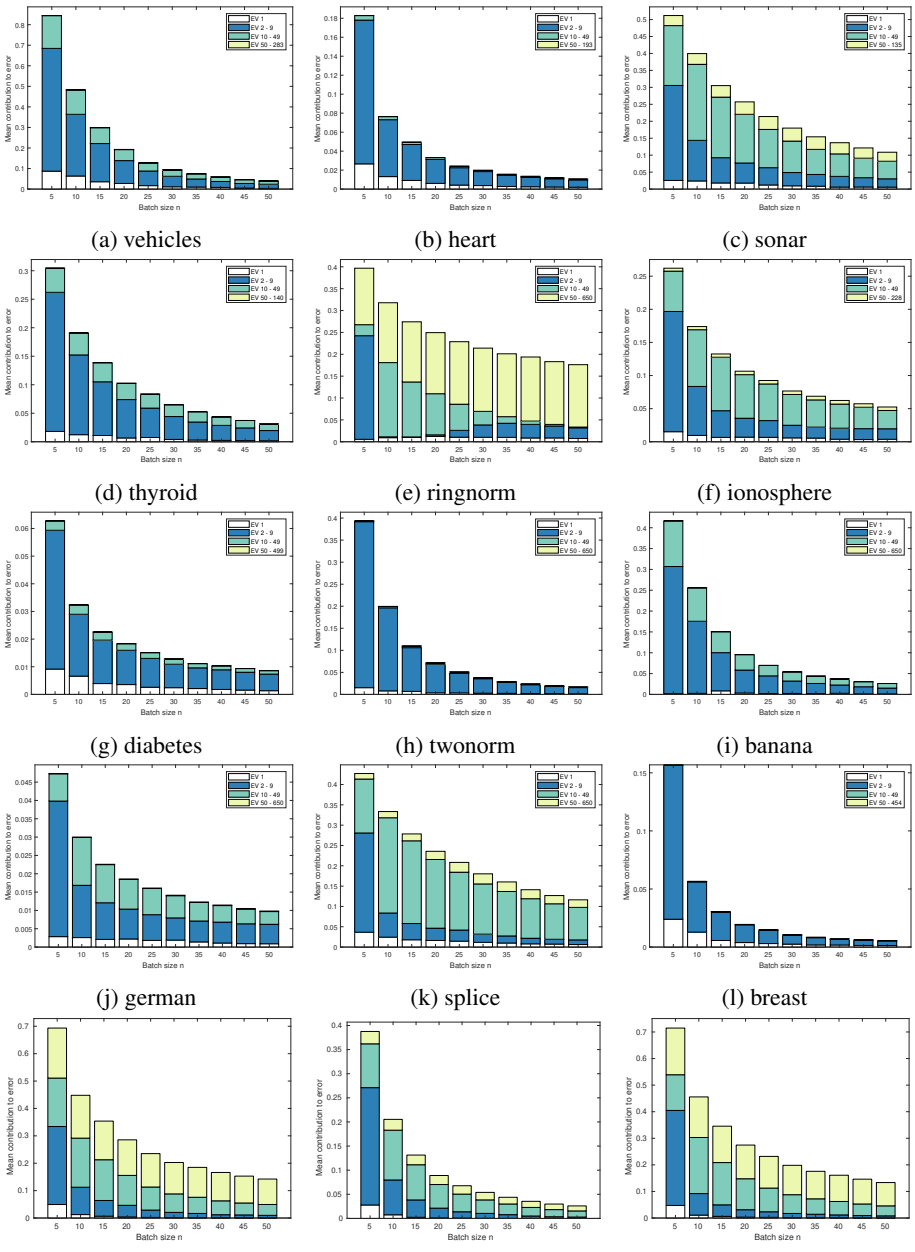
(j) german (k) splice (l) breast

(m) mnist 3vs5 (n) mnist 7vs9 (o) mnist 5vs8

Figure C.3.: Results on all benchmark datasets for the realizable setting.

C

C.8.2. DECOMPOSITION OF THE PROBABILISTIC BOUNDS FOR ALL DATASETS



(a) vehicles (b) heart (c) sonar

(d) thyroid (e) ringnorm (f) ionosphere

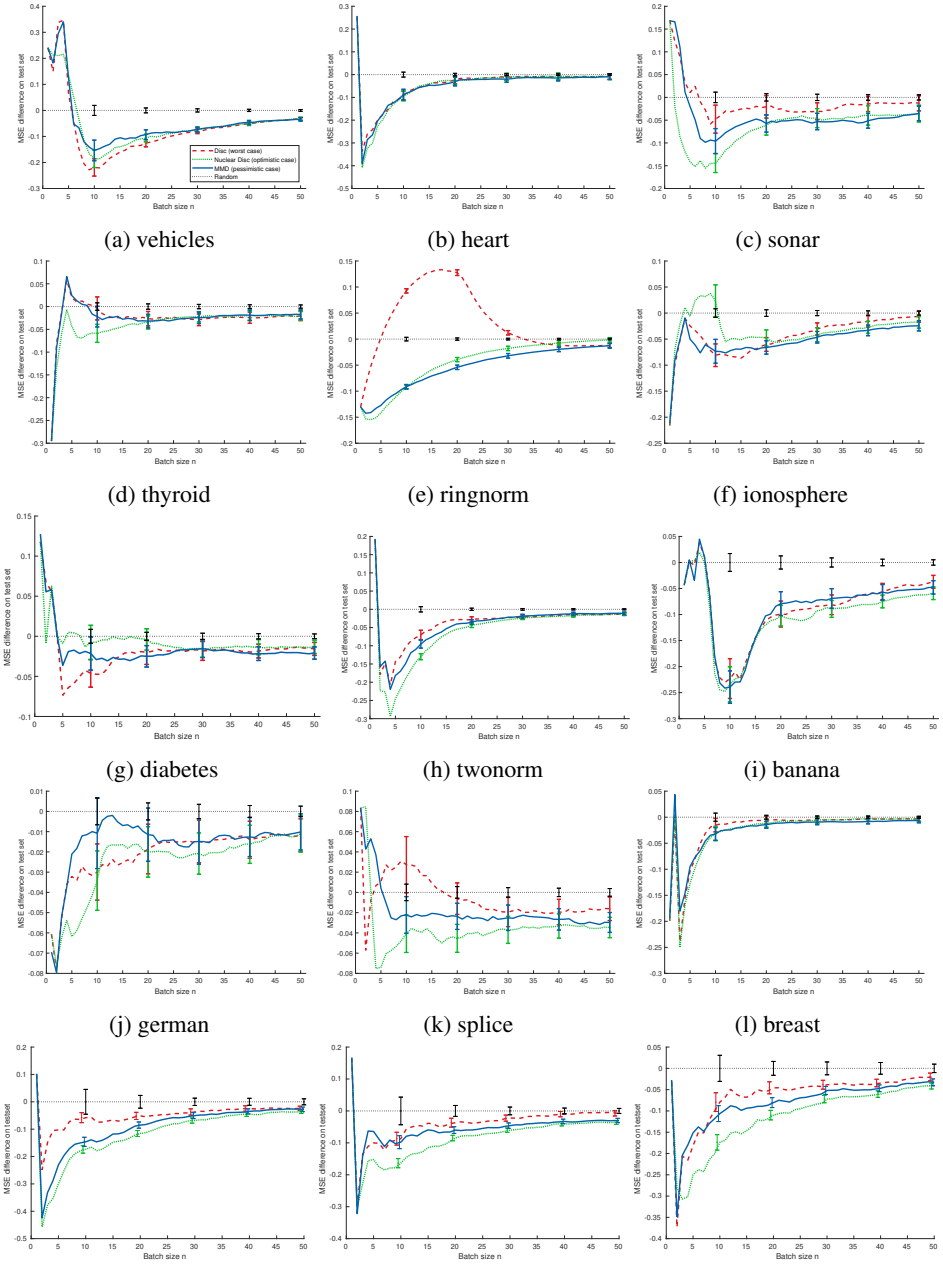
(g) diabetes (h) twonorm (i) banana

(j) german (k) splice (l) breast

(m) mnist 3vs5 (n) mnist 7vs9 (o) mnist 5vs8

Figure C.4.: Decomposition of the sum $G(u, M)$ during active learning for all datasets for the active learner 'random sampling'.

C.8.3. LEARNING CURVES ON ALL DATASETS FOR THE AGNOSTIC SETTING



(a) vehicles (b) heart (c) sonar

(d) thyroid (e) ringnorm (f) ionosphere

(g) diabetes (h) twonorm (i) banana

(j) german (k) splice (l) breast

(m) mnist 3vs5 (n) mnist 7vs9 (o) mnist 5vs8

Figure C.5.: Results on all benchmark datasets for the agnostic setting.

C.9. BIBLIOGRAPHY

[1] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning* (MIT press, Cambridge, Massachusetts, 2012).

[2] Y. Mansour, M. Mohri, and A. Rostamizadeh, *Domain Adaptation: Learning Bounds and Algorithms,* in *Proceedings of the 22nd Annual Conference on Learning Theory (COLT)* (2009).

[3] C. Cortes and M. Mohri, *Domain adaptation and sample bias correction theory and algorithm for regression,* Theoretical Computer Science **519**, 103 (2014).

[4] C. Cortes, M. Mohri, and A. M. Medina, *Adaptation based on generalized discrepancy,* Journal of Machine Learning Research **20**, 1 (2019).

[5] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis* (Cambridge University Press, Cambridge, UK, 2004).

[6] G. C. Cawley and N. L. Talbot, *Fast exact leave-one-out cross-validation of sparse least-squares support vector machines,* Neural Networks **17**, 1467 (2004).

[7] M. Lichman, *UCI Machine Learning Repository,* (2013).

[8] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, *Gradient-based learning applied to document recognition,* Proceedings of the IEEE **86**, 2278 (1998).

D

Appendix of Chapter 5

D.1. Theorem 8

The minimizing hypothesis of the empirical risk $A_{\text{erm}}(S_n)$ is attained for the mean that equals $\mu_n := \frac{1}{n}\sum_{i=1}^{n} z_i$. Let $\mathbb{E}$ denote the expectation both over the training sample $S_n \sim P^n$ and over the test sample z. The expected negative log-likelihood for μ_n then equals

$$\begin{aligned} &-\mathbb{E}\left[\log\left(\frac{1}{\sqrt{(2\pi)^d|\Sigma|}}\exp(-\tfrac{1}{2}(z-\mu_n)^T\Sigma^{-1}(z-\mu_n))\right)\right] \\ &= -\log\left(\frac{1}{\sqrt{(2\pi)^d|\Sigma|}}\right) + \tfrac{1}{2}\mathbb{E}\left[(z-\mu_n)^T\Sigma^{-1}(z-\mu_n)\right]. \end{aligned} \tag{D.1}$$

Only the last term differs for different training set sizes n, so we only need to consider that part when studying the monotonicity of the learning curve. Taking μ to be the true first moment (and dropping the $\frac{1}{2}$), we can rewrite the last term as

$$\begin{aligned} &\mathbb{E}\left[(z-\mu_n)^T\Sigma^{-1}(z-\mu_n)\right] = \mathbb{E}\left[(z-\mu+\mu-\mu_n)^T\Sigma^{-1}(z-\mu+\mu-\mu_n)\right] \\ &= \mathbb{E}\left[(z-\mu)^T\Sigma^{-1}(z-\mu)\right] + \mathbb{E}\left[(\mu_n-\mu)^T\Sigma^{-1}(\mu_n-\mu)\right]. \end{aligned} \tag{D.2}$$

Again, it is only the last term that matters as it is the only part that differs for different training set sizes. For this term, we have that

$$\begin{aligned} &\mathbb{E}\left[(\mu_n-\mu)^T\Sigma^{-1}(\mu_n-\mu)\right] = \mathbb{E}\left[(\tfrac{1}{n}\sum_{i=1}^{n} z_i-\mu)^T\Sigma^{-1}(\tfrac{1}{n}\sum_{i=1}^{n} z_i-\mu)\right] \\ &= \mathbb{E}\left[(\tfrac{1}{n}\sum_{i=1}^{n} z_i)^T\Sigma^{-1}(\tfrac{1}{n}\sum_{i=1}^{n} z_i)\right] - \mu^T\Sigma^{-1}\mu \\ &= \mathbb{E}\left[\tfrac{1}{n^2}\sum_{i=1}^{n} z_i^T\Sigma^{-1}z_i + \tfrac{1}{n^2}\sum_{j\neq k} z_j^T\Sigma^{-1}z_k\right] - \mu^T\Sigma^{-1}\mu \\ &= \mathbb{E}\left[\tfrac{1}{n^2}nz^T\Sigma^{-1}z + \tfrac{1}{n^2}(n^2-n)\mu^T\Sigma^{-1}\mu\right] - \mu^T\Sigma^{-1}\mu \\ &= \mathbb{E}\left[\tfrac{1}{n}z^T\Sigma^{-1}z\right] - \tfrac{1}{n}\mu^T\Sigma^{-1}\mu = \tfrac{1}{n}\mathbb{E}\left[(z-\mu)^T\Sigma^{-1}(z-\mu)\right]. \end{aligned} \tag{D.3}$$

As the domain is bounded, $\mathbb{E}\left[(z-\mu)^T\Sigma^{-1}(z-\mu)\right]$ exists and we can readily conclude that the expected negative log-likelihood decreases with increasing n. Therefore, A_{erm} is globally monotonic. □

D.2. LEMMA 2

Let $P(a) = q$ and $P(b) = 1-q$. The expected risk over S_n then equals

$$R(q) := \sum_{k=0}^{n} \binom{n}{k} q^k (1-q)^{n-k} \left(q\ell(a, h^k_{n-k}) + (1-q)\ell(b, h^k_{n-k}) \right). \tag{D.4}$$

The derivative to q of the above equals

$$\begin{aligned}\frac{d}{dq}R(q) = \sum_{k=0}^{n} \binom{n}{k} \Big[&(k+1)q^k(1-q)^{n-k}\ell(a, h^k_{n-k}) - (n-k)q^{k+1}(1-q)^{n-k-1}\ell(a, h^k_{n-k}) + \\ &kq^{k-1}(1-q)^{n-k+1}\ell(b, h^k_{n-k}) - (n-k+1)q^k(1-q)^{n-k}\ell(b, h^k_{n-k}) \Big].\end{aligned} \tag{D.5}$$

Taking the limit $q \to 0$, all terms become zero for $k > 1$. For $k = 0$, we get $\ell(a, h^0_n) - (n+1)\ell(b, h^0_n)$ and, for $k = 1$, we get $n\ell(b, h^1_{n-1})$. Similarly, for a training sample size of $n+1$, the only nonzero terms we get are for $k \in \{0, 1\}$, as the expression for the derivative is essentially the same.

It shows that the q-derivative evaluated in 0 of the difference in expected risk from Equation (5.4) equals $\ell(a, h^0_{n+1}) - (n+2)\ell(b, h^0_{n+1}) + (n+1)\ell(b, h^1_n) - \ell(a, h^0_n) + (n+1)\ell(b, h^0_n) - n\ell(b, h^1_{n-1})$, which can be further simplified to $-\ell(b, h^0_{n+1}) + (n+1)\ell(b, h^1_n) - n\ell(b, h^1_{n-1})$, as $\ell(a, h^0_n) = \ell(a, h^0_{n+1})$ and $\ell(b, h^0_n) = \ell(b, h^0_{n+1})$.

If this derivative is strictly larger than 0, continuity in q implies that there is a $q > 0$ such that the actual risk difference becomes positive. This shows that A_{erm} is not locally $(\mathcal{Z}, \ell, n)$-monotonic. □

D.3. THEOREM 9

Let us first consider the squared loss. Take $a = (a_1, 0, \ldots, 0, a_{d+1})$ and $b = (b_1, 0, \ldots, 0, b_{d+1})$, such that the input vectors, $(a_1, 0, \ldots, 0)$ and $(b_1, 0, \ldots, 0)$, which constitute the first d coordinates are in $B_0 \subset \mathcal{Z}$. The variables a_{d+1} and b_{d+1} constitute the outputs. Let both first input coordinates a_1 and b_1 not be equal to 0. All other input coordinates do equal 0. In this case, all (minimum-norm) hypotheses are finite and Remark 2 applies to this setting. So we study whether $(n+1)\ell(b, h^1_n) - n\ell(b, h^1_{n-1}) > 0$ in order to be able to invoke Lemma 2. To do so, we exploit that we can determine h^1_n in closed-form. As all input variation occurs in the first coordinate only, we have that $h^1_n = \left(\frac{a_1 a_{d+1} + n b_1 b_{d+1}}{a_1^2 + n b_1^2}, 0, \ldots, 0 \right) \in \mathbb{R}^d$, which implies that $\ell(b, h^1_n) = \left(b_1 \frac{a_1 a_{d+1} + n b_1 b_{d+1}}{a_1^2 + n b_1^2} - b_{d+1} \right)^2$. In the same way we, find that $\ell(b, h^1_{n-1}) = \left(b_1 \frac{a_1 a_{d+1} + (n-1) b_1 b_{d+1}}{a_1^2 + (n-1) b_1^2} - b_{d+1} \right)^2$. Now take the limit of b_1 to 0 to obtain $(n+1)\ell(b, h^1_n) - n\ell(b, h^1_{n-1}) = (n+1)b^2_{d+1} - n b^2_{d+1} = b^2_{d+1}$. For any b_{d+1} bounded

away from 0, this shows that for all $n \in \mathbb{N}$ there is a $b_1 > 0$, small enough, such that $(n+1)\ell(b, h^1_n) - n\ell(b, h^1_{n-1}) > 0$. This shows in turn that there exists a b_1 and a corresponding $b_{d+1} \neq 0$, such that A_{erm} under the squared loss is not locally $(\mathcal{Z}, \ell, n)$-monotonic. As this holds for all n, we conclude that it also is not weakly $(\mathcal{Z}, \ell, N)$-monotonic for any $N \in \mathbb{N}$.

For the absolute loss, we consider the same setting as for the squared loss and its very beginning proceeds along the exact same lines. The proof starts to deviate at the calculation of $\ell(b, h^1_n)$ and $\ell(b, h^1_{n-1})$. Still the same as for the squared loss, as all input variation occurs in the first coordinate, we only have to study what happens in that subspace. This means that all other $d-1$ elements of the minimum-norm solutions we consider will be 0. As h^1_n is the empirical risk minimizer for one a and n bs, we have

$$h^1_n = \operatorname*{argmin}_{h \in \mathbb{R}^d} \frac{1}{n+1} \left(|a_1 h_1 - a_{d+1}| + n|b_1 h_1 - b_{d+1}|\right), \tag{D.6}$$

where h_1 is the first element of h. We can rewrite the main part of the objective function as

$$|a_1 h_1 - a_{d+1}| + n|b_1 h_1 - b_{d+1}| = |a_1| \left| h_1 - \frac{a_{d+1}}{a_1} \right| + n|b_1| \left| h_1 - \frac{b_{d+1}}{b_1} \right|. \tag{D.7}$$

From this, one readily sees that the first coordinate of the minimizer h^1_n equals $\frac{a_{d+1}}{a_1}$ if $|a_1| > n|b_1|$ and $\frac{b_{d+1}}{b_1}$ if $|a_1| < n|b_1|$. If $|a_1| = n|b_1|$, then it picks $\min(\frac{a_{d+1}}{a_1}, \frac{b_{d+1}}{b_1})$ as we are looking for the minimum-norm solution. For that same reason, all other entries of h^1_n equal 0. Similar expressions, with $n-1$ substituted for n, hold for h^1_{n-1}. If we take $|b_1| < \frac{|a_1|}{n+1}$, then we get $(n+1)\ell(b, h^1_n) - n\ell(b, h^1_{n-1}) = (n+1)|\frac{a_{d+1}}{a_1} b_1 - b_{d+1}| - n|\frac{a_{d+1}}{a_1} b_1 - b_{d+1}| = |\frac{a_{d+1}}{a_1} b_1 - b_{d+1}|$, which is larger than 0 if $a_1 b_{d+1} \neq b_1 a_{d+1}$. Again along the same lines as for the squared loss, this shows that regression using the absolute loss is not locally $(\mathcal{Z}, \ell, n)$-monotonic and, as this holds for all n, we conclude that it is not weakly $(\mathcal{Z}, \ell, N)$-monotonic for any $N \in \mathbb{N}$.

Finally, the hinge loss. As we are necessarily dealing with a classification setting now, a_{d+1} and b_{d+1} are in $\{-1, +1\}$. Now, take $a_1 > 0$, $b_1 > 0$, $a_{d+1} = +1$ and $b_{d+1} = -1$. Any choice of h can only classify either a or b correctly, as both a_1 and b_1 are positive. With this, the empirical risk of h^1_{n+1} becomes $\frac{1}{n+1}(\max(0, 1 - a_1 h) + n \max(0, 1 + b_1 h))$ and only solutions h for which the first coordinate is in $[-\frac{1}{b_1}, \frac{1}{a_1}]$ need to be considered, as values outside of this interval will only increase the loss for either a or b, while the loss remains the same for the other values. Being limited to the interval $[-\frac{1}{b_1}, \frac{1}{a_1}]$ implies $\max(0, 1 - a_1 h) = 1 - a_1 h = |1 - a_1 h|$. As we have a similar loss in point b, we will find exactly the same solutions as we found for the absolute loss, but with a_{d+1} and b_{d+1} limited to $\{-1, +1\}$. □

D.4. THEOREM 10

Take a and b to be in $B_0 \subset \mathcal{Z}$. As opposed to the proof for Theorem 9, we now cannot use the suggestion from Remark 2, as for the log-likelihood it does not hold that $\ell(b, h^0_n) = \ell(b, h^0_{n+1}) = 0$. Therefore, we need to look at the full expression of Lemma 2: $-\ell(b, h^0_{n+1}) + (n+1)\ell(b, h^1_n) - n\ell(b, h^1_{n-1})$. The sigma that belongs to the empirical risk

minimizing hypothesis h^0_{n+1} equals $\sqrt{b^2}$. For h^1_{n-1} it is $\sqrt{\frac{a^2+(n-1)b^2}{n}}$ and for h^1_n we get $\sqrt{\frac{a^2+nb^2}{n+1}}$. Therefore, we come to the following *negative* log-likelihoods:

$$\ell(b, h^0_{n+1}) = \log|b| + \frac{1}{2} + \frac{1}{2}(\log(2) + \log(\pi)), \tag{D.8}$$

$$\ell(b, h^1_n) = \frac{nb^2}{2\left(a^2 + (n-1)b^2\right)} + \log\left(\sqrt{\frac{a^2 + b^2(n-1)}{n}}\right) + \frac{1}{2}(\log(2) + \log(\pi)), \tag{D.9}$$

$$\ell(b, h^1_{n-1}) = \frac{(n+1)b^2}{2\left(a^2 + nb^2\right)} + \log\left(\sqrt{\frac{a^2 + b^2 n}{n+1}}\right) + \frac{1}{2}(\log(2) + \log(\pi)). \tag{D.10}$$

D

Now, consider the limit of b going to 0. The last two negative log-likelihoods are finite in that case, while $\ell(b, h^0_{n+1})$ will go to minus infinite. This implies that for $b > 0$ small enough, we have that $-\ell(b, h^0_{n+1}) + (n+1)\ell(b, h^1_n) - n\ell(b, h^1_{n-1}) > 0$ (because of the term $-\ell(b, h^0_{n+1})$). In conclusion, our density estimator is not locally $(\mathcal{Z}, \ell, n)$-monotonic and, as this holds for all n, we conclude that it is not weakly $(\mathcal{Z}, \ell, N)$-monotonic for any $N \in \mathbb{N}$. □

E

Exact Learning Curve Distribution for Wrapper

We give a procedure to calculate the learning curve distribution for the wrapper algorithm $\mathrm{MT}_{\mathrm{SIMPLE}}$, which we will apply to Example 7.1 (p. 146).

Let W_{i+1}, W_i be the random variables of the true error rate of the wrapper in round $i+1$ and i, respectively. Let L_i be the random variable that corresponds to the true error rate of the original learner $A(S_i)$ in round i to which the wrapper algorithm is applied. The corresponding model is called h_i. Since the training set S_i that is supplied to the learner is a random sample, L_i is a random variable. The decisions of the wrapper algorithm depend on the values of L_i and on randomness coming from the validation sample, therefore W_i is also a random variable.

We make the (rather strong) assumption that the probability density function of L_i is known for all i, as is the case in Example 7.1 (p. 146). We further assume that all L_i are independent, which can be achieved in general by assuming that the training datasets S_1, S_2, etc. are always collected from scratch. Under these assumptions we will give a procedure to compute the probability density function of W_i for all i for $\mathrm{MT}_{\mathrm{SIMPLE}}$ in case the validation set size $N_v = 1$.

In each round except the first, $\mathrm{MT}_{\mathrm{SIMPLE}}$ compares the previously best model h_{best} to the newly trained model h_i. While not explicitly mentioned in Chapter 6, this comparison should be pessimistic, in the sense that, if the models tie, the wrapper will use the previously best model h_{best} [1]. As we have seen in that chapter, pessimism is necessary to make the curve more monotone. Since we have only one validation sample, this comes down to *only* choosing the new model h_i if that model makes no mistake on the validation sample while the previously best model h_{best} *does*. See Table E.1 for an overview of the situation.

We could in addition, define a random variable B_i that indicates the error rate of the previous best model h_{best}. Note, however, due to the recursive nature of the wrapper algorithm, we have that $B_{i+1} = W_i$ for all rounds $i > 1$. In other words, the previously best model in round $i+1$ can be obtained by running the wrapper algorithm i rounds. The model comparison made by the wrapper in round $i+1$, is thus comparing models whose error rates are

[1] In Chapter 6 we always set $N_v \gg 1$, in which case such ties are very rare.

		h_{best} (error rate W_i)	
	loss on validation sample	0 (correct)	1 (incorrect)
h_i (error rate L_{i+1})	0 (correct)	choose h_{best}	choose h_i
	1 (incorrect)	choose h_{best}	choose h_{best}

Table E.1.: An overview of the action of the wrapper algorithm in round $i+1$, depending on the different zero one losses of h_{best} and h_i on the *single* validation sample.

given by W_i and L_{i+1}.

In the first round, $\text{MT}_{\text{SIMPLE}}$ always returns $h_1 = A(S_1)$, since there is no previously best model available to compare with. Thus, $W_1 = L_1$, and since we assumed the density of all L_i are known, we know the density of W_1.

Now we will recursively compute the density of W_{i+1} using the densities W_i and L_{i+1}. Note that, L_{i+1} and W_i are always independent, since h_{i+1} uses training set S_{i+1}, while the model corresponding to the error rate W_i has only had access to the training sets $S_1, \ldots, S_i$. Using independence and the rule of total probability,

$$P(W_{i+1} < a) = P(W_{i+1} < a | W_i < a, L_{i+1} < a) P(W_i < a) P(L_{i+1} < a) \tag{E.1}$$
$$+ P(W_{i+1} < a | W_i \geq a, L_{i+1} \geq a) P(W_i \geq a) P(L_{i+1} \geq a) \tag{E.2}$$
$$+ P(W_{i+1} < a | W_i \geq a, L_{i+1} < a) P(W_i \geq a) P(L_{i+1} < a) \tag{E.3}$$
$$+ P(W_{i+1} < a | W_i < a, L_{i+1} \geq a) P(W_i < a) P(L_{i+1} \geq a) \tag{E.4}$$

The last two probabilities of each line can readily be calculated by integrating the known probability density functions of L_{i+1} and W_i. Thus let us now discuss how to compute the first probability of each line.

We have $P(W_{i+1} < a | W_i < a, L_{i+1} < a) = 1$ (Equation E.1), since here the choice of the wrapper algorithm does not matter, the wrapper will always achieve an error rate smaller than a regardless of which model is chosen. Using a similar argument we have $P(W_{i+1} < a | W_i \geq a, L_{i+1} \geq a) = 0$ (Equation E.2), thus we can ignore the second line.

For $P(W_{i+1} < a | W_i \geq a, L_{i+1} < a)$ (Equation E.3), the wrapper needs to choose h_i, which *only* occurs if h_i makes no mistake on the validation sample while h_{best} does (the top right event in Table E.1). For the case $N_v = 1$, conditioning on $L_{i+1} = l$ and $W_i = w$, the probability of that event is $w(1-l)$, where we can multiply the probabilities due to independence. However, W_i and L_{i+1} are random variables, and thus we need to integrate out their probability density functions to obtain the definitive probability for this event. Note that we are computing a probability which is conditioned on $W_i \geq a, L_{i+1} < a$, thus the densities should also be conditioned on this. Making use of the fact that $N_v = 1$ we obtain

$$\int_{w \in [a,1]} \int_{l \in [0,a]} w(1-l) f_{W_i | W_i \geq a}(w) f_{L_{i+1} | L_{i+1} < a}(l) \, dw \, dl$$

where $f_{W_i | W_i \geq a}(w)$ is the conditional probability density of W_i conditioned on $W_i \geq a$. We use the same notation for the conditional density of L_{i+1}. Simplifying we find

$$P(W_{i+1} < a | W_i \geq a, L_{i+1} < a, N_v = 1) = \mathbb{E}(W_i | W_i \geq a)(1 - \mathbb{E}(L_{i+1} | L_{i+1} < a)). \tag{E.5}$$

For $P(W_{i+1} < a | W_i < a, L_{i+1} \geq a)$ (Equation E.4), h_{best} needs to be chosen, which is exactly the complementary event (see Table E.1). Thus for the case $N_v = 1$, conditioning on $L_{i+1} = l$ and $W_i = w$, the probability of that event is $1 - w(1-l)$. Integrating out the densities we find for the case $N_v = 1$ that

$$\int_{w \in [0,a]} \int_{l \in [a,1]} (1 - w(1-l))\, f_{W_i | W_i < a}(w) f_{L_{i+1} | L_{i+1} \geq a}(l)\; dw\; dl$$

simplifying further we obtain

$$P(W_{i+1} < a | W_i < a, L_{i+1} \geq a, N_v = 1) = 1 - \mathbb{E}(W_i | W_i < a)(1 - \mathbb{E}(L_{i+1} | L_{i+1} \geq a)). \quad \text{(E.6)}$$

Now we have completely expressed $P(W_{i+1} < a)$ in Equation E.1 in terms that only depend on integrals of W_i and L_{i+1}. Differentiating $P(W_{i+1} < a)$ with respect to a, we find the probability density function of W_{i+1}. This can be recursively applied to obtain all W_i.

Let us illustrate the procedure using Example 7.1 (p. 146). For this example, the probability density function for all L_i is uniform on the interval $[0, 1]$. Thus we also have that W_1 is uniform on $[0, 1]$ since always $L_1 = W_1$. Let us now compute W_2, which depends on L_2 and W_1. Note that, in this special case, we have that $L_2 = W_1$, and thus some calculations simplify. First, let us turn to the last two probabilities of each line in the Equations E.1 - E.4. By integrating the uniform distribution we find $P(W_1 < a) = P(L_2 < a) = a$ and $P(W_1 \geq a) = P(L_2 \geq a) = 1 - a$. Using the definition of the conditional density function we find that

$$f_{W_1 | W_1 < a}(w) = \frac{1}{a},$$

$$f_{W_1 | W_1 \geq a}(w) = \frac{1}{1-a},$$

note that these conditional densities are only valid on their respective domains, otherwise they are zero, e.g. $f_{W_1 | W_1 < a}(w) = 0$ for $w > a$. From these we can compute conditional expectations

$$\mathbb{E}(W_1 | W_1 < a) = \frac{a}{2},$$

$$\mathbb{E}(W_1 | W_1 > a) = \frac{1}{2} + \frac{a}{2},$$

and using Equations E.5 and E.6 we find

$$P(W_2 < a | W_1 \geq a, L_2 < a) = \left(1 - \frac{a}{2}\right)\left(\frac{a}{2} + \frac{1}{2}\right),$$

$$P(W_2 < a | W_1 < a, L_2 \geq a) = \frac{a}{2}\left(\frac{a}{2} - \frac{1}{2}\right) + 1.$$

Putting everything together in Equations E.1 - E.4 and simplifying we eventually find that

$$P(W_2 < a) = \frac{a}{2}(3 - a).$$

Now we differentiate this cumulative density function to obtain

$$f_{W_2}(a) = \frac{3}{2} - a,$$

and so we have obtained the probability density of W_2 from the densities L_2 and W_1.

The procedure is thus: compute $P(W_{i+1} < a)$ from L_{i+1} and W_i using Equation E.1 - E.4 and differentiating to obtain $f_{W_{i+1}}$. We can recursively keep applying this to compute all the densities for W_i for arbitrary i. Let us give the third and fourth density analytically here:

$$f_{W_3}(a) = \frac{1}{2}a^2 - \frac{13}{6}a + \frac{23}{12}$$

$$f_{W_4}(a) = -\frac{1}{4}a^3 + \frac{19}{12}a^2 - \frac{251}{72}a + \frac{41}{18}$$

The exact means can be readily computed from these densities and are for $i = 1,\ldots,4$ given by $\frac{1}{2}$, $\frac{5}{12}$, $\frac{13}{36}$, $\frac{697}{2160}$. We give a picture to illustrate the distribution in Figure 7.2 (right) on page 147 for $i = 1,\ldots,30$.

ACKNOWLEDGEMENTS

First of all, I am very grateful to be a part of the research group PRB. It is difficult for me to imagine a more stimulating group, and I'm very glad to be a part of it. While the Thursday night beers hooked me as an MSc student, the group activities and friendly atmosphere is what makes me want to stick around. Hats off to **Marcel** for continuing to make sure this research group continues running so smoothly.

Second, I want to thank my closest collaborator and office roommate, **Alexander**. Our pre-lunch discussions will be a fond memory of my PhD. I am forever indebted to you for your patience when helping me out with the homework for that machine learning theory course. Most importantly you have taught me that math isn't that hard, you just have to stick with it. Finally, thanks for teaching me how to lose some weight :).

Marco, first of all, thanks for providing me the opportunity to do a PhD with you. You have helped me understand what great research is *really* about. I will never forget to ask 'why' ever again; and now I understand the mathematicians approach to research. One of the important lessons you have taught me is to be more selective in what to do. Thanks for all your support and the slight pushes that helped me finish the PhD relatively on time. Without it surely it would still not be finished. Thanks for all the great times we had inside and outside the office. I am looking forward to more fruitful collaborations and all other sorts of mischief in the future!

Elmar, I am really sad that most of our collaborations did not pan out as we planned them. However, thanks for running a nice research group and always making me feel welcome. Furthermore, I want to thank you for your critical reading of my thesis.

Amogh, I fondly remember those few times you were in Delft. Luckily, my Amsterdam visits have made up for this. I hope we can keep having these research visits that involve, for example, the NDSM or the Greek islands. Of course, also thanks to your significant other **Neha**, who always manages to raise my spirits.

Speaking of more partying... **Taylan** and **Rickard**, it was a pleasure to meet you and I am looking forward to our next raves. Of course, I cannot forget **Ziqi**. I think our collaboration, where we wrote our related work section in Croatia while drinking cocktails, will be hard to beat. Further international collaborations are fortunately not ruled out in London.

Talking about trips, I want to thank all other next door neighbors, **Chirag**, **Jose**, **Yeshwanth**, (and also **Ziqi**), with whom I went on a euro trip. I absolutely had a blast. **Yeshwanth** thanks for all the driving, your great spirits and nice cooking. Besides that, I have to mention my trip with **Chirag** and **Sonakshi** to Sziget which was too much fun, thanks again! Sonakshi, thanks for the good times and introducing me to your awesome friends. I am looking forward to go again to Sziget to meet **Jakob** & **Kamilla** again. Oh, and I'm sorry **Yancong** for making you stay so long on that boat trip in Amsterdam, but I think it was cool you went along! **Yunqiang**, I will also fondly remember our trip to BMVC where we had many memorable moments together with **Ziqi**.

Laura, Ekin, Wouter, Sally, Christian, I think you guys really made such a great atmosphere in the group. **Sally**, thanks for some wise words and that awesome Mexican BBQ. I will never forget our Ibiza trip as dolphin trainers with **Laura** and **Alexander**. **Laura**, thanks for teaching me the ways of making awesome PhD movies. **Wouter**, thanks for all the memes and your high spirits. **Ekin**, thanks for teaching me vaping and other weird stuff.

Hayley, thanks for now and then that motivating cookie. **Jose** and **Stephanie**, thanks for the discussions and insightful talks. **Laura**, thanks for trying to teach us some hardware stuff. **Bernd**, thanks for all the cool talks as well. **Chirag**, its unfortunate our collaboration never got off the ground, but I hope we can work together in the future! **Tiffany**, I hope we will work less late from now on...! Thanks for the good company!

Ruud, thanks for all the smoke breaks and more than once saving me with Linux! **Bart** and **Robbert**, also thanks for more than once helping me with the cluster. **Saskia** and **Marunka**, thanks for all your work to keep the group running smoothly.

Osman, thanks for all the entertaining and excellent talks and all your positivity! **Silvia**, I always cherished our (too short) travel time together back towards Leiden where you always had some wise words for me. **Nergis**, thanks for all the smoke breaks! **Seyran**, thanks for all the tasty treats. Special thanks to **Hamdi**. You always had some great advice for me and supported me in my journey to quit smoking. **Wenjie**, I also enjoyed our talks, I hope you are doing well in China. **Robert-Jan, Atillia, Xucong, Xin, Xiangwei, Ombretta, Marian,** thanks for interesting discussions and the great atmosphere.

Jan, thanks for running such a great research group and organizing such a useful website. I want to thank you in particular for presenting so many things patiently that others take for granted. Keep it up, you are an asset to the group.

Bio people; **Stavros**, I greatly enjoyed our brainstorming sessions and the fruitful student projects that came from it. **Thomas**, thanks for offering me the opportunity to supervise a Bio student together. **Christine**, thanks for all the laughter. **Aysun, Ramin, Yasin, Skander,** thanks for all enjoyable activities together and the good talks!

Arman, thanks for all the great (smoke) breaks and I am sure we will have some more fun in the future visiting China. **Jin, Ramin, Mahdi, and Mo,** thanks for all the company. **Yazhou**, thanks for now and then helping me out with my active learning research. **Taygun**, thanks for the interesting discussions. I also want to thank the numerous MSc and BSc students of the PR Lab who joined during the PR meetings and the drinks, such as **Berend**, **Prajit**, **Aleksander** and **Pia**. **Yuko** and **Taylan**, thanks for helping me out with the Capstone. I am hoping for more collaborations in the future!

David, as a teacher you were a great inspiration to me as you always manage to excite your audience. I hope we at some point will work together teaching. **Jesse**, thanks again for pushing me during my Msc thesis. But also during my PhD I want to thank you for more than once for some wise words! **Bob**, thanks for many interesting talks.

Now besides being part of PRB, I was also lucky enough to be part of the teaching team as a lecturer. Many thanks to **Andy**, who coached me through developing a whole course on my own (also thanks to **David**, **Jesse** and **Neil**!). **Andy**, I want to thank you further for stimulating me to apply for the new position in Delft, and all your other mentoring and support! Many thanks to **Neil** and **Przemek**, for all things relating to organizing the AI minor, the machine learning course and the Capstone. **Gosia**, thanks for all the fun we've had, and I'm looking forward to our future collaborations! Special thanks to **Susanne van**

Aardenne for our collaboration on the surf proposal and the pitch. **Wendy**, thanks for all the laughs we have shared, and the opportunities you have provided me. **Panchamy**, many thanks for organizing the FAIP course where I had the pleasure of giving multiple lectures!

Many thanks to all people in the teaching team and their endurance of me complaining about the Capstone. Many thanks to **Frank**, for all the laughs, but also for distilling the most important points of the UTQ into just a few Zoom meetings. Also many thanks to **Bart**: for the beers and tips and tricks with Vocareum. **Thomas**, thanks for all the sweets, beers, but most of all, all your support and input for organizing the Capstone. **Otto**, thanks for the help with all sorts of things such as the digital rubric, and all the beers! **Stefan**, thanks for getting me up to speed with all kinds of things, such as Zesje. **Taico**, thanks for all kinds of IT support, especially with project forum. Furthermore, thanks to all other members of the teaching team for the great atmosphere!

Furthermore, I also greatly enjoyed some of the integrating activities with the research group of Interactive Intelligence, such as collaborations with **Pradeep, Frank and Arkady** (the latter actually works in AiTech). Furthermore, I was very pleased to meet **Elena**, who interestingly I got to know much better at BMVC in Belgium than in Delft. I am confident you will do great in the teaching team and will easily wrap up your thesis. I'm looking forward to more adventures! **Zuza**, thanks for organizing all kinds of fun activities! **Davide, Ruben, Enrico, Rolf, Sandy, Mani,** great to meet you!

Also thanks to **Jan van Rijn** for many enjoyable activities in Leiden and our great collaboration with **Felix**. Special thanks to **Marie** for many inspirations in my research. I was also very lucky to meet **Jan and Christina Gopfert**, and I hope we will soon meet again! Also sincere apologies to **Zak Mhammedi** who we kept awake with our open problem on monotone learning. I am happy it has led you to work on this problem! Thanks to **Peter Grünwald** for the machine learning theory course and the lecture at TU Delft.

Then I would like to thank **Irene, Quincy, Marieke (aka zoon), Ruby, Suki, Else, Pieter, Nina**. Thanks for all the great parties and I'm sure we will see eachother soon enough again at Liquicity! **Liquicity**, thanks for organizing such wonderful parties, please keep it up! **Wallie**, thanks for all the fun in Amsterdam! I am looking forward to our next party. Thanks to **Ineke**, who has offered me a lot of support and some great distractions during my PhD.

Jelmer, thanks for all the advice and all the beers! Your support and mentoring has always greatly helped me, and I'm looking forward to our future adventures. **Ivar**, thanks for all the fun with the boardgames (which you always win)! **Tom Vieveen**, thanks for the good times! **Frank Marsman, Leandros, Jeroen, Rembrand** thanks for all the great times and parties! **Mihail** thanks for hosting us so many times in Amsterdam and showing us around in Greece! Also thanks to **Olivier, Bas, Arnold, David,** for the beers and fun! Also many thanks to all the **Pongols (Katwijk, Freek, Koen, Maarten, Nigel, Oli)** for all the fun times and not removing me from the Whatsapp group :). Thanks also to **Karlien, Natasja, Luigi and Oli** for all the great meetups!

Tenslotte, wil ik graag mijn ouders bedanken. **Alma**, bedankt voor alle steun en wijze adviezen, en je altijd scherpe blik. **Peter**, bedankt voor alle gezelligheid, koffies bij Roos en vrolijkheid. Jullie hebben me altijd gestimuleerd om me verder te ontwikkelen en het beste uit mezelf te halen, heel erg bedankt hiervoor!

CURRICULUM VITÆ

Tom Julian Viering

14-01-1991	Born in Leiden, Netherlands.

EDUCATION

2003-2009	High School,
	Rijnlands Lyceum, Oegstgeest
2009-2013	Bachelor Physics,
	Leiden University
2013-2016	Master in Computer Science,
	Delft University of Technology
2016-2023	Ph.D. in Computer Science
	Delft University of Technology

Thesis:	On Safety in Machine Learning
Promotors:	Prof. dr. E. Eisemann
	Prof. dr. M. Loog

LIST OF PUBLICATIONS

13. M. Loog, **T. J. Viering**, *A Survey of Learning Curves with Bad Behavior: or How More Data Need Not Lead to Better Performance*, accepted at Benelearn, Mechelen, 2022.

12. P. Bhaskaran, **T. J. Viering**, *To Tune or not to Tune: Hyperparameter Influence on the Learning Curve*, accepted at Benelearn, Mechelen, 2022 (student paper).

11. D. Kim, **T. J. Viering**, *Different approaches to fitting and extrapolating the learning curve*, accepted at Benelearn, Mechelen, 2022 (student paper).

10. F. Mohr, **T. J. Viering**, M.Loog, J. N. van Rijn, *LCDB 1.0: An Extensive Learning Curves Database for Classification Tasks*, accepted at European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD), Grenoble, 2022.

9. **T. J. Viering**, M. Loog, *The Shape of Learning Curves: a Review*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022 (TPAMI).

8. K. G. Schmahl, **T. J. Viering**, S. Makrodimitris, A. Naseri Jahfari, D. Tax, M. Loog, *Is Wikipedia succeeding in reducing gender bias? Assessing changes in gender bias in Wikipedia using word embeddings*, Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science (NLPCSS), pages 94-103, Virtual Conference, 2020 (student paper).

7. M. Loog, **T. J. Viering**, A. Mey, J. H. Krijthe, D. M. Tax, *A Brief Prehistory of Double Descent*, Proceedings of the National Academy of Sciences (PNAS), 117(20): 10625-10626, 2020.

6. A. Mey, **T. J. Viering**, M. Loog, *A Distribution Dependent and Independent Complexity Analysis of Manifold Regularization*, International Symposium on Intelligent Data Analysis (IDA), pages 326-338, Virtual Conference, 2020.

5. **T. J. Viering**, A. Mey, M. Loog, *Making Learners (More) Monotone*, International Symposium on Intelligent Data Analysis (IDA), pages 535-547, Virtual Conference, 2020.

4. **T. J. Viering**, Z. Wang, M. Loog, E. Eisemann, *How to Manipulate CNNs to Make Them Lie: the Gradcam Case*, Proceedings of the British Machine Vision Conference (BMVC) 2019 Workshop on Interpretable and Explainable Machine Vision (IXMV), Cardiff, UK, 2019.

3. M. Loog, **T. J. Viering**, A. Mey, *Minimizers of the Empirical Risk and Risk Monotonicity*, Advances in Neural Information Processing Systems (NeurIPS) 32, pages 7476-7485, Vancouver, Canada, 2019.

2. **T. J. Viering**, A. Mey, M. Loog, *Open Problem: Monotonicity of Learning*, Conference on Learning Theory (COLT), pages 3198-3201, Phoenix, Arizona, 2019.

1. **T. J. Viering**, J. H Krijthe, M. Loog, *Nuclear Discrepancy for Single-Shot Batch Active Learning*, Machine Learning, 108(8-9): 1561-1599, 2019 (ECMLPKDD).