LEARNING CURVES WITH LITTLE DATA

Oğuz Taylan Turan

Learning Curves with Little Data

Learning Curves with Little Data

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. H. Bijl,

chair of the Board for Doctorates

to be defended publicly on

Tuesday 7 July 2026 at 17:30

by

Özgür Taylan TURAN

This dissertation has been approved by the (co)promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. ir. M.J.T. Reinders,	Delft University of Technology, *promotor*
Prof. dr. M. Loog,	Radboud University, *promotor*
Dr. D.M.J. Tax,	Delft University of Technology, *copromotor*
Independent members:	
Prof. dr. D. Gavrila	Delft University of Technology
Dr. J.C. van Gemert	Delft University of Technology
Prof. dr. M. van Leeuwen	Leiden University
Dr. T.M. van Laarhoven	Radboud University
Prof. dr. F.A. Oliehoek	Delft University of Technology, *reserve member*

Keywords: learning curve, generalization performance, meta-learning, data-scarcity

Printed by: Proefschriftspecialist

Cover by: Özgür Taylan Turan

ISBN 978-94-6384-981-4

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

In loving memory of Nurten and Tugay Turan.
For the time I could not spend with you.

Contents

Summary

Obtaining data is often costly, making it important to assess whether collecting additional data is justified by the expected improvement in performance. Learning curves, which describe the expected performance of a learner as a function of dataset size, provide a useful tool for this purpose. They can help practitioners assess whether further data collection is justified by the anticipated gains. However, additional data does not always lead to improved performance, which makes estimating the potential benefit challenging. Under such conditions, model selection also becomes more difficult, as it is unclear how to compare models when no data is available to evaluate their performance at a hypothetical training set size.

In this context, the thesis takes a step back and asks a more fundamental question: how can we reliably reason about generalization when data is scarce and the behavior of learning curves is itself uncertain? Rather than treating learning curves as simple, and monotonic functions, we study their full statistical structure. We show that variability across training subsets can influence model comparison, decision making, and performance extrapolation. In addition, we investigate conditions under which monotonic improvement can be guaranteed or encouraged. Beyond single task learning, we also examine meta-learning, where information from multiple related tasks is leveraged to improve generalization performance while reducing the amount of data required from any individual task.

We begin by showing that the mean, as a statistical summary of learning curves, may not provide a reliable estimate of performance. We demonstrate that generalization performance distributions are often skewed and heavy tailed, regardless of how they are obtained. As a result, relying solely on the mean for model selection can be suboptimal for some problems.

Next, we propose a semi parametric extrapolation method that adapts its inductive bias to capture complex and potentially non monotonic patterns. This approach improves predictive reliability in settings where additional data collection is costly or infeasible and where learning curves may not exhibit monotonic behavior.

We then study the monotonicity of learning curves under specific conditions. For linear regression, we show that a single gradient update is sufficient to ensure monotonic improvement, provided that the learning rate does not exceed a certain threshold. To construct similarly monotonic learners in practice, we propose a data driven approach for selecting both the learning rate and the initial parameter estimates.

Finally, we investigate the learning curves of a meta learning algorithm. Through controlled synthetic experiments, we analyze the generalization performance of both meta learners and task specific learners, providing insights into how properties of the task distribution influence generalization under a limited adaptation stage consisting of a single gradient update.

Samenvatting

Het verkrijgen van data is vaak kostbaar, waardoor het belangrijk is om te beoordelen of het verzamelen van data gerechtvaardigd is door de verwachte verbetering van de prestaties. Leercurves, die de verwachte prestatie van een model beschrijven als functie van de grootte van de dataset, vormen hiervoor een nuttig hulpmiddel. Ze kunnen praktijkbeoefenaars helpen te beoordelen of verdere dataverzameling gerechtvaardigd is door de verwachte winst. Extra data leidt echter niet altijd tot betere prestaties, wat het moeilijk maakt om de potentiële voordelen in te schatten. Onder dergelijke omstandigheden wordt ook modelselectie complexer, omdat het onduidelijk is hoe modellen met elkaar vergeleken moeten worden wanneer er geen data beschikbaar is om hun prestaties te evalueren bij een hypothetische trainingssetgrootte.

In deze context doet de thesis een stap terug en stelt een meer fundamentele vraag: hoe kunnen we op betrouwbare wijze redeneren over generalisatie wanneer data schaars is en het gedrag van leercurves zelf onzeker is? In plaats van leercurves te beschouwen als eenvoudige en monotone functies, bestuderen we hun volledige statistische structuur. We laten zien dat variabiliteit tussen trainingssubsets modelvergelijking, besluitvorming en prestatie-extrapolatie kan beïnvloeden. Daarnaast onderzoeken we onder welke omstandigheden monotone verbetering kan worden gegarandeerd of gestimuleerd. Naast single task learning bestuderen we ook meta-learning, waarbij informatie uit meerdere verwante taken wordt benut om de generalisatieprestatie te verbeteren en tegelijkertijd de hoeveelheid benodigde data per individuele taak te verminderen.

We beginnen met aan te tonen dat het gemiddelde, als statistische samenvatting van leercurves, mogelijk geen betrouwbare schatting van de prestaties geeft. We laten zien dat verdelingen van generalisatieprestaties vaak scheef en zwaarstaartig zijn, ongeacht hoe ze worden verkregen. Als gevolg daarvan kan het uitsluitend vertrouwen op het gemiddelde voor modelselectie bij sommige problemen suboptimaal zijn.

Vervolgens stellen we een semi-parametrische extrapolatiemethode voor die haar inductieve bias aanpast om complexe en mogelijk niet-monotone patronen te kunnen vastleggen. Deze benadering verbetert de voorspellende betrouwbaarheid in situaties waarin aanvullende dataverzameling kostbaar of niet haalbaar is en waarin leercurves mogelijk geen monotoon gedrag vertonen.

Daarna bestuderen we de monotonie van leercurves onder specifieke voorwaarden. Voor lineaire regressie laten we zien dat één enkele gradiëntupdate voldoende is

om monotone verbetering te garanderen, mits de leersnelheid een bepaalde drempel niet overschrijdt. Om in de praktijk vergelijkbaar monotone modellen te construeren, stellen we voor gebruik te maken van een datagedreven aanpak voor het selecteren van zowel de leersnelheid als de initiele parameterschattingen.

Tot slot onderzoeken we de leercurves van een meta-learningalgoritme. Door middel van gecontroleerde synthetische experimenten analyseren we de generalisatieprestaties van zowel meta-leerders als taakspecifieke leerders, en bieden we inzicht in hoe eigenschappen van de taakverdeling de generalisatie beïnvloeden onder een beperkte adaptatiefase die bestaat uit een enkele gradiëntupdate.

1 | Introduction

One of the most frequently used terms in this dissertation is *"learning"*. Throughout this dissertation it is used to refer the process of induction: drawing general rules from limited examples. In other words, upon observing apples and pears, learning refers to the process of finding the rule or function that separates the two.

Humans rely on this type of inductive reasoning constantly, often without noticing it. Imagine a friend who is late to every meeting. You might infer that this friend will always be late. This inference is not guaranteed to be correct, but it is a reasonable expectation given the data you have.

Machines can also be designed to perform inductive reasoning [1, 2]. Given a hypothesis space and a metric for evaluating correctness, a machine can be programmed to find *a* hypothesis, not necessarily *the* hypothesis. The found hypothesis explains the observed data with regards to the selected metric. For example, a machine might classify an apple as edible not because it recognizes it as an apple and it has some edible properties, but because the example provided was a red spherical object. Consequently, a red ball might also be classified as edible under the same learned rule.

The way we use the term *learning* for algorithms aligns perfectly with Mitchell's definition [3]: a computer program, or machine, is said to learn if its performance on a task, such as distinguishing elephants from rhinos, improves with increasing experience. In this definition; experience refers to data, for instance it can be characteristic features of elephants and rhinos, and one hopes that as the number of examples of elephants and rhinos increase a capable learning algorithm finds rule(s) such as, if a trunk or tusk is present, it is an elephant, and if a horn is present, it is a rhino. The fundamental assumption is that such a rule exists and data, or examples, support this rule.

What happens when we cannot increase the number of examples, though? Indeed considering the amount of data that we are exposed to from all the social media channels, it is easy to assume that data is always abundant, but this is not always the case. Let us consider the example of classifying rhinos again, but this time we want to distinguish between two different rhino species, namely, Sumatran and Javan rhinos, both of which are endangered. At the time of writing this dissertation, there are around fifty (hopefully at the time of you reading it is more) rhinos from both species [4]. Although a small set of examples may allow a rule that fits the available examples, the challenge lies in learning a rule that generalizes beyond the observed examples.

With so few rhino examples, the found rule may fail to capture the natural variability within each species. As a result, obtaining a general rule for distinguishing between two endangered rhino species that have not yet been observed is more difficult than in cases where larger populations are available.

Data-scarcity effects various domains from engineering [5] to healthcare [6]. Experiments and measurements often rely on specialized equipment, long-term monitoring or expensive labeling all of which limit scalability. In engineering fields there are computational tools to circumvent expensive or timely experimentations, such as determining the humidity and temperature behaviour of materials used in wind turbines [7]. Even these computational counter parts of the resources required for a particular problem can consume considerable time and resources.

When the data is scarce, there are two main questions to address: how much of it is needed to reach a desired performance level, and, if obtaining more data is not possible, how the generalization performance of our models can be improved. The former question is addressed through learning curves [8], introduced in the following Section 1.1. The latter can be investigated under the umbrella of learning-to-learn, which is introduced in Section 1.2 .

1.1 Learning Curves

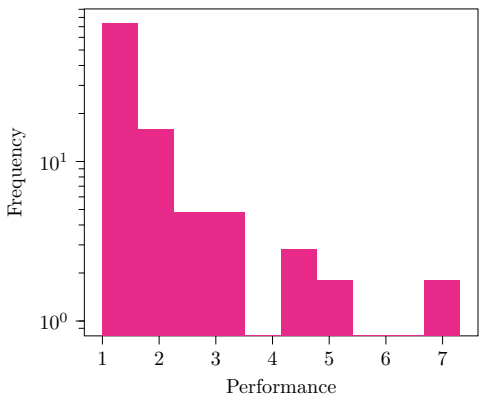
The term *learning curve* is multivalent. In the artificial neural network literature it often refers to generalization performance as a function of the number of training iterations (epochs) [9]. This is also called a *training curve* [10]. In the broader machine learning field, however, a learning curve typically expresses generalization performance with respect to the number of training examples. In [11], a learning curve is defined a bit more broadly as the performance of a learning algorithm with respect to some parameter of the learning process, such as epochs, complexity of the model or training set size. This unification can be problematic because, depending on the variable under investigation curves represent different notions of generalization performance and are obtained in different ways. Therefore, unless stated otherwise, *learning curve* in this dissertation refers specifically to generalization performance as a function of training set size.

An example of a learning curve is shown in Figure 1.1a in green. It is computed by averaging the individual generalization performances, shown as scatter points in the same figure, of a learning algorithm on a held-out testing data. These performance measurements are obtained from the models trained on sampled subsets of the overall training data.

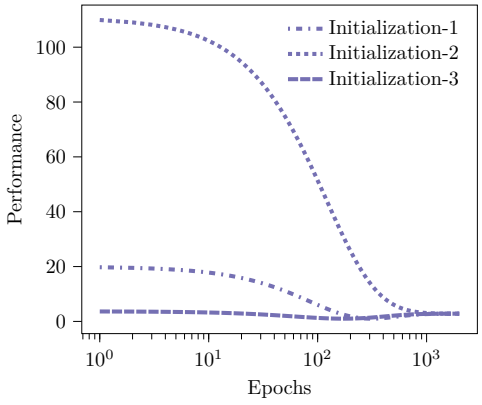
The distinction between a learning curve and a training curve becomes clearer in the plot highlighted in Figure 1.1c. In this plot, the training curve shows the gen-

eralization performance for three different random initializations, evaluated at every update as the learning algorithm follows the gradient of the loss over the dataset. The circled point on the learning curve corresponds to the final point of one of these training curves, trained on one randomly selected training set. This example illustrates the difference between learning curves and training curves.

(a) Illustrative learning curve.



(b) Performance distribution histogram at the highlighted point in Figure 1.1a.



(c) Sample training curves for the highlighted training set size in Figure 1.1a.

Figure 1.1: Learning curve vs training curve for a one-dimensional linear regression problem with Mean Squared Error as the performance metric.

Observing the overall trajectory of performance with respect to training set size provides practitioners with information about a learning algorithm's generalization

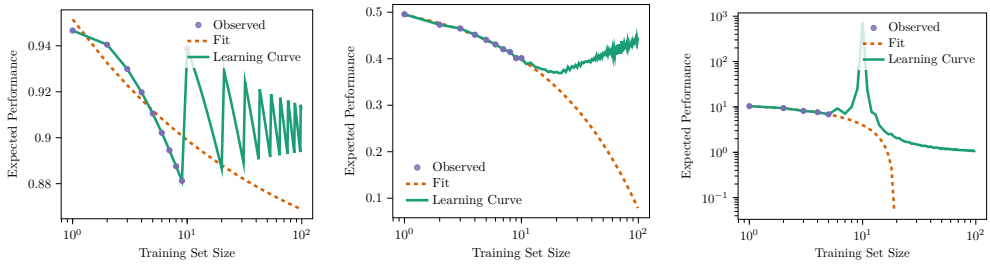
performance on that specific dataset. For instance, in Figure 1.1a the learners is not gaining much more generalization after 10 training points. This suggests further increases in training set size unlikely to yield benefit and may motivate exploring alternative model classes if better performance is expected. However, using averages, the green line in Figure 1.1a, as statistical summaries for learning curves can be misleading, as the performance may appear overly optimistic due to a particularly favorable sampled training subset, or unrealistically poor due to an unfavorable one. The probability of such extreme outcomes increases in data-scarce regimes. Figure 1.1b illustrates a histogram of generalization performance at a fixed training set size on the learning curve. The distribution shows that different sampled training subsets can lead to quite distinct performance outcomes.

Perhaps more importantly, learning curves help determine whether collecting additional data is worthwhile. A stable expected performance can indicate that further data collection is unlikely to yield meaningful improvements in generalization performance. This is a critical consideration in data-scarce settings where data acquisition is the primary bottleneck. To estimate potential performance gains from additional data, we face a fundamental challenge: the portion of the learning curve we wish to examine has not yet been observed. For instance, in practice, we may not have access to the shaded region in Figure 1.1a. To gain insight into expected generalization performance in these regions, learning curves must be extrapolated. Historically, this relies on parametric models with built-in assumptions of monotonicity and saturation [10]. These assumptions imply that performance improves with more data and eventually plateaus. While monotonicity is often motivated by the intuition that more data leads to better generalization, it rests on other assumptions about model capacity, data distributions, and the loss measures.

It is therefore not surprising that these assumptions may fail: more data is not always better [12, 13, 14, 15, 16]. In [10] a taxonomy of non-monotone curves are presented, where different reasons for monotonicity, such as modeling choice, data distribution, objective mismatch between training and testing loss, are identified as some of the causes of non-monotonic behaviour. If additional data is costly and difficult to obtain, determining the performance gain from additional data becomes even more important compared to the case of a monotonic curve. Some examples of non-monotonic learning curves are shown in Figure 1.2.

1.1.1 Monotonicity of Learning Curves

Because adding more data does not always lead to better results, a natural question is: when can we be sure that having more training examples will actually improve performance? Several theoretical studies have identified settings in which this guarantee



(a) Sawing curve for absolute difference.

(b) Dipping curve for error rate.

(c) Peaking curve for mean squared error.

Figure 1.2: Some examples of non-monotonic learning curves for different performance measures (in green), presented in [10], are shown together with power-law-with-offset fits (orange dashed lines) to the observed points on the learning curve (blue scatter points).

holds. For example, [17] shows that in a classification problem, where performance is measured by accuracy, performance improves as more data is added, provided that the set of possible prediction rules is limited in both size and complexity. Similarly, [18] present two problems for which performance improves monotonically as additional data become available. The first is fitting a normal distribution with fixed covariance and an unknown mean. The second is a specialized algorithm that is able to memorize the observed inputs.

Beyond identifying conditions under which learning performance improves with more data, it is also practically important to design methods that enforce such monotonic behavior. In this direction, [19] proposes an algorithm that guarantees, with high probability, monotonic improvement in accuracy. The key idea is to discard models that violate monotonicity on a validation set when additional data is introduced. This approach is further refined by [20], who improve monotonic behavior by explicitly constraining the difficulty of the learning problem. Moreover, [21] shows that, in multi-class classification, any classifier can be transformed into a monotonic counterpart without sacrificing accuracy. Finally, [22] demonstrates that in regression problems, selecting an optimal level of regularization ensures monotonic improvement in performance as more data is added.

Although the above mentioned efforts of finding when a learner is monotonic or how the monotonicity can be ensured, it is still an open problem, under unbounded losses, such as mean squared error (MSE), the existence of a monotone learner is yet to be discovered.

1.1.2 Extrapolating Learning Curves

To extrapolate learning curves one has to determine the shape of the learning curve. In [23], it is suggested that the learning curves of deep learning models follow a power-law. However, in [12] a large learning-curve database is created and analysed, where a wide set of learning algorithms are considered. All learning curves are fitted with a wide range of parametric models and the best performing ones are found to be more flexible than a power-law.

In Figure 1.2, we present power-law (with offset) fits to the limited portion of the learning curve data. The unreasonable extrapolations produced by these fits to limited data and its effect on the extrapolation capabilities are evident across various types of non-monotonic curves. [13] also presents similar findings where the non-monotone curves are more difficult to extrapolate. To tackle this, one approach is division of the learning curve into regions and each region is predicted by again parametric formulations [24], assuming piece-wise monotonicity.

Then, how can we extrapolate learning curves in the presence of non-monotonicity? One way of fitting complex functions is using non-parametric models, however, if the data is extremely limited non-parametric modeling is difficult (due to their degrees of freedom). Another way can be to actually study a large amount of learning curves to try to find out better parametric forms that can represent real non-monotonic learning curves. However, maybe a more practical approach is to create data-driven ways that are learning from a large number of learning curves directly. This approach is recently gaining attention with foundation models that can learn to extrapolate in data-scarce settings. One such example is presented recently in [25] where a large collection of previously observed learning curves is used to train a foundation model that can extrapolate learning curves upon seeing little data from the beginning of the curve. However, the problem of extrapolating non-monotonic learning curves remains open.

1.2 Learning-to-Learn

Data-scarce settings are referred to as small-sample size problem in the pattern recognition field [26, 27], and nowadays, in the machine/deep learning field introduced as few-shot learning. The challenge in this regime is that the data for a single task are insufficient for induction [28]. A variety of strategies exist for enriching limited data with auxiliary meta-information. These include data augmentation, generating synthetic observations, or leveraging related tasks or domains [6].

The setting where the meta-information comes from other related tasks is coined as meta-learning or learning-to-learn [29]. This closely mimics some settings in which humans learn. For instance, [30] shows that previous tasks can aid learning a related task. Going even further, [31] shows that even if the previous tasks are not directly

related, some early stages of the learning are more efficient.

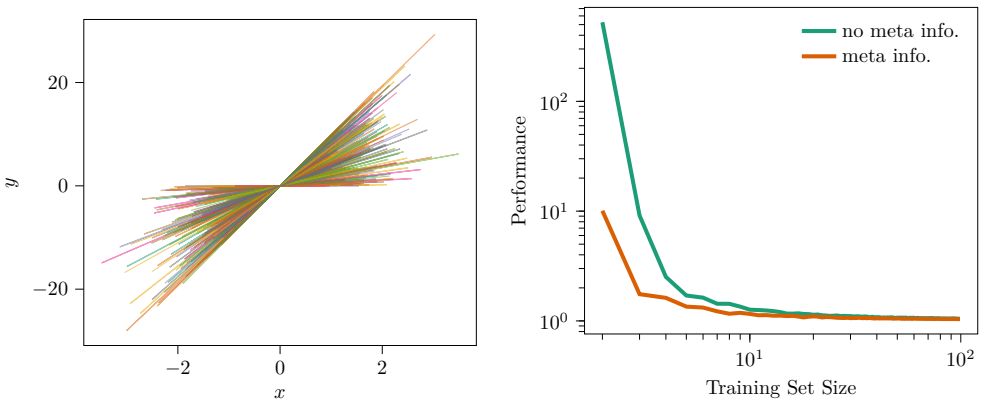
Meta-learning, nowadays, is often viewed through the lens of modern deep learning, particularly following the influential work of [32], it has become widely recognized for enabling models to adapt quickly to new tasks [29]. However, the concept itself has much deeper historical roots. In classical machine learning contexts, Mitchell's definition [3] can be used for this paradigm in a similar spirit: a computer system learns to learn when, given examples from related tasks, its performance improves as it encounters additional tasks and accumulates experience [33]. Crucially, this definition does not emphasize rapid adaptation in terms of fine-tuning time, but rather data-efficient learning, achieving better performance with fewer examples. This suggests that exposure to related tasks reduces the overall data requirements for learning new tasks.

To make this idea more concrete, we consider a family of related regression problems drawn from a common task distribution. Representative datasets from this family are shown in Figure 1.3a. In each task, the underlying data-generating process is a linear function whose slope varies across tasks, while all lines are constrained to pass through the origin.

Our goal is to estimate the slope from noisy observations of a task drawn at random and to evaluate the resulting generalization performance. To highlight the role of meta-information, we assume access to multiple datasets sampled from the task distribution in Figure 1.3a. From these, we extract shared structural knowledge, specifically that all tasks correspond to functions that pass through the origin. This information is then incorporated explicitly into the model used for a new, previously unseen task.

Figure 1.3b compares the learning curves of two linear regression models evaluated using mean squared error. The first model exploits this meta-information by enforcing the zero-intercept constraint, while the second model is trained without any such prior knowledge. The advantage of incorporating meta-information is particularly evident in the low-data regime, where the performance gap between the two approaches is largest, demonstrating improved generalization from fewer samples.

In this thesis, meta-learning refers to the process of extracting a transferable structure across tasks to improve learning efficiency on new tasks. In the example mentioned above, the process of obtaining priors from other tasks is what we mean by meta-learning. However, it can also involve learning common temporal patterns, such as spikes, trends, or periodic behaviors, for a regression problem too. With this additional information, the hope is that for an unseen task, the model can recognize similar patterns and make more accurate predictions with only a small amount of data. Rather than optimizing a model solely on samples from a single task, meta-learning operates on a distribution of tasks and aims to learn an initialization, a representation,



(a) Multiple linear regression tasks that are all going through the origin $(0, 0)$.

(b) Learning curves comparing performance with and without meta-information.

Figure 1.3: Effect of incorporating simple meta-information on the expected performance of the linear regression setting.

or relevant information that allows the model to adjust its inductive bias on-the-fly when only a few new samples are available.

Since meta-learning aims to improve the ability to learn in data-scarce settings, it is an attractive research direction with several open challenges. In [28], these challenges include the definition of realistic task distributions, the need for more diverse and representative benchmarking datasets, and developing theoretical understanding of meta-learners.

1.3 Contributions

In Chapter 2, we investigate the generalization performance distributions shown in Figure 1.1b. Generally, while estimating the generalization performance, practitioners utilize a limited number of re-samplings of the training set. Consequently, the resulting trained performances are averaged over a low number of sampling and only sometimes the standard deviation is considered when performing model selection or hyper-parameter tuning. We investigate the performance distributions along learning curves and show the implications of looking at the average of performances in comparison to other statistical measures.

In Chapter 3, we tackle the problem of extrapolating non-monotone learning curves, which are typically observed under specific data distributions and loss measures. We introduce a semi-parametric learning curve extrapolation method capable of handling this non-monotonicity by automatically adjusting its inductive bias.

Linear Regression also shows a non-monotonic learning curve when data is scarce.

When the number of data points are smaller than the features for Ordinary Least Squares (OLS), the minimum-norm solution results in a non-monotonic behaviour. In Chapter 4, we present a setting where a linear model can have a monotone learning curve. Furthermore, we propose a method that can be applied in practical problems to make learning curves of linear regression more monotone.

In Chapter 5, we analyze the generalization performance of the meta-learner MAML [32] and a set of task-specific learners with respect to the parameters governing the task distribution, using a synthetic experimental setting.

Chapter 2 and 3 requires collecting large numbers of learning curves. We present a learning curve generation tool in Appendix D which enables reproducible learning curve generation with or without hyperparameter tuning with various options for training set sampling and can be scaled easily high-performance computing environments.

1

References

[1] R. J. Solomonoff. "A Formal Theory of Inductive Inference. Part I". In: *Information and Control* 7.1 (Mar. 1964), pp. 1–22. DOI: 10.1016/S0019-9958(64)90223-2.

[2] R.J. Solomonoff. "A Formal Theory of Inductive Inference. Part II". In: *Information and Control* 7.2 (June 1964), pp. 224–254. DOI: 10.1016/S0019-9958(64)90131-7.

[3] Tom M. Mitchell. *Machine Learning*. Nachdr. McGraw-Hill Series in Computer Science. New York: McGraw-Hill, 2013.

[4] World Wildlife Fund. *Rhinoceros*. https://www.worldwildlife.org/species/rhino. 2025.

[5] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations". In: *Journal of Computational Physics* 378 (Feb. 2019), pp. 686–707. DOI: 10.1016/j.jcp.2018.10.045.

[6] Fabian Gröger et al. "A Review and Systematic Guide to Counteracting Medical Data Scarcity for AI Applications". In: *Computer Methods and Programs in Biomedicine Update* 8 (Jan. 2025), p. 100220. DOI: 10.1016/j.cmpbup.2025.100220.

[7] I.B.C.M. Rocha. "Numerical and Experimental Investigation of Hygrothermal Aging in Laminated Composites". Dissertation (TU Delft). 2019. DOI: 10.4233/uuid:0eab23c7-9ba4-4d27-91ee-58f9f140dd34.

[8] Rafid Mahmood et al. "Optimizing Data Collection for Machine Learning". In: *Journal of Machine Learning Research* 26.38 (2025), pp. 1–52.

[9] Claudia Perlich. "Learning Curves in Machine Learning". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 577–580. DOI: 10.1007/978-0-387-30164-8_452.

[10] Tom Viering and Marco Loog. "The Shape of Learning Curves: A Review". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.6 (2023), pp. 7799–7819. DOI: 10.1109/TPAMI.2022.3220744.

[11] Felix Mohr and Jan N. van Rijn. "Learning Curves for Decision Making in Supervised Machine Learning: A Survey". In: *Machine Learning* 113.11 (Dec. 2024), pp. 8371–8425. DOI: 10.1007/s10994-024-06619-7.

[12] Felix Mohr et al. “LCDB 1.0: An Extensive Learning Curves Database for Classification Tasks”. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Massih-Reza Amini et al. Cham: Springer Nature Switzerland, 2023, pp. 3–19.

[13] Cheng Yan, Felix Mohr, and Tom Viering. *LCDB 1.1: A Database Illustrating Learning Curves Are More Ill-Behaved Than Previously Thought*. 2025. arXiv: 2505.15657 [cs.LG].

[14] Robert PW Duin. “Small sample size generalization”. In: *Proceedings of the Scandinavian Conference on Image Analysis*. Vol. 2. 1995, pp. 957–964.

[15] Marco Loog and Robert P. W. Duin. “The Dipping Phenomenon”. In: *Structural, Syntactic, and Statistical Pattern Recognition*. Ed. by Georgy Gimel’farb et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 310–317.

[16] Marco Loog, Jesse H. Krijthe, and Manuele Bicego. “Also for K-Means: More Data Does Not Imply Better Performance”. In: *Machine Learning* 112.8 (Aug. 2023), pp. 3033–3050. DOI: 10.1007/s10994-023-06361-6.

[17] Ming Li, Chenyi Zhang, and Qin Li. “Monotonic Learning in the PAC Framework: A New Perspective”. In: *Knowledge-Based Systems* 330 (Nov. 2025), p. 114504. DOI: 10.1016/j.knosys.2025.114504.

[18] Marco Loog, Tom Viering, and Alexander Mey. “Minimizers of the empirical risk and risk monotonicity”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[19] Tom Julian Viering, Alexander Mey, and Marco Loog. “Making Learners (More) Monotone”. In: *Advances in Intelligent Data Analysis XVIII*. Ed. by Michael R. Berthold, Ad Feelders, and Georg Krempl. Cham: Springer International Publishing, 2020, pp. 535–547.

[20] Zakaria Mhammedi. “Risk Monotonicity in Statistical Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021.

[21] Olivier J Bousquet et al. “Monotone Learning”. In: *Proceedings of Thirty Fifth Conference on Learning Theory*. Ed. by Po-Ling Loh and Maxim Raginsky. Vol. 178. Proceedings of Machine Learning Research. PMLR, Feb. 2022, pp. 842–866.

[22] Preetum Nakkiran et al. *Optimal Regularization Can Mitigate Double Descent*. Apr. 2021. arXiv: 2003.01897 [cs, math, stat].

[23] Yasaman Bahri et al. “Explaining neural scaling laws”. In: *Proceedings of the National Academy of Sciences* 121.27 (June 2024). DOI: 10.1073/pnas.2311878121.

[24] Ethan Caballero et al. “Broken Neural Scaling Laws”. In: *The Eleventh International Conference on Learning Representations*. 2023.

[25] Tom Julian Viering et al. “From Epoch to Sample Size: Developing New Data-driven Priors for Learning Curve Prior-Fitted Networks”. In: *AutoML Conference 2024 (Workshop Track)*. 2024.

[26] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Computer science and scientific computing. Academic Press, 2013.

[27] A.K. Jain and B. Chandrasekaran. “39 Dimensionality and Sample Size Considerations in Pattern Recognition Practice”. In: *Handbook of Statistics*. Vol. 2. Elsevier, 1982, pp. 835–855. DOI: 10.1016/S0169-7161(82)02042-2.

[28] Anna Vettoruzzo et al. “Advances and Challenges in Meta-Learning: A Technical Review”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.7 (July 2024), pp. 4763–4779. DOI: 10.1109/TPAMI.2024.3357847.

[29] Pavel Brazdil et al. *Metalearning: Applications to Automated Machine Learning and Data Mining*. Cognitive Technologies. Cham: Springer International Publishing, 2022. DOI: 10.1007/978-3-030-67024-5.

[30] R. S. Woodworth and E. L. Thorndike. “The Influence of Improvement in One Mental Function upon the Efficiency of Other Functions. (I).” In: *Psychological Review* 8.3 (May 1901), pp. 247–261. DOI: 10.1037/h0074898.

[31] Florian Kattner et al. “Perceptual Learning Generalization from Sequential Perceptual Training as a Change in Learning Rate”. In: *Current Biology* 27.6 (Mar. 2017), pp. 840–846. DOI: 10.1016/j.cub.2017.01.046.

[32] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1126–1135.

[33] Sebastian Thrun and Lorien Pratt, eds. *Learning to Learn*. Boston, MA: Springer US, 1998. DOI: 10.1007/978-1-4615-5529-2.

2 | Generalization Performance Along Learning Curves

Learning curves show the expected performance with respect to training set size. This is often used to evaluate and compare models, tune hyper-parameters and determine how much data is needed for a specific performance. However, the distributional properties of performance are frequently overlooked on learning curves. Generally, only an average with standard error or standard deviation is used. In this paper, we analyze the distributions of generalization performance on the learning curves. We compile a high-fidelity learning curve database, both with respect to training set size and repetitions of the sampling for a fixed training set size. Our investigation reveals that generalization performance rarely follows a Gaussian distribution, regardless of dataset balance, loss function, sampling method, or hyper-parameter tuning along learning curves. Furthermore, we show that the choice of statistical summary, mean versus measures like quantiles affect the top model rankings. Our findings highlight the importance of considering different statistical measures and use of non-parametric approaches when evaluating and selecting machine learning models with learning curves.

2.1 Introduction

In (supervised) machine learning, the goal is often to optimize the expected performance of a model. For optimizing the expected performance, samples are assumed to be drawn from some fixed distribution. However, in practice, we do not have access to this distribution. Instead, we rely on training and testing sets obtained from finite datasets, which introduces stochasticity. Factors such as, initialization or optimization procedures, further contribute to the variability of model performance.

To account for stochasticity, performance is most often summarized using the average [1]. While averages are useful in many problems, they are not always sufficient. If the performance distribution is highly skewed, exhibits heavy tails, or contains substantial outliers, the mean may fail to capture the underlying behavior. In such cases, more robust estimators such as quantiles can provide a clearer picture. For example, in high-risk settings, one may prefer a more conservative or, conversely, a more opti-

Published as: **O. Taylan Turan** et al. "Generalization Performance Distributions Along Learning Curves". In: Pattern Recognition Letters (2026).

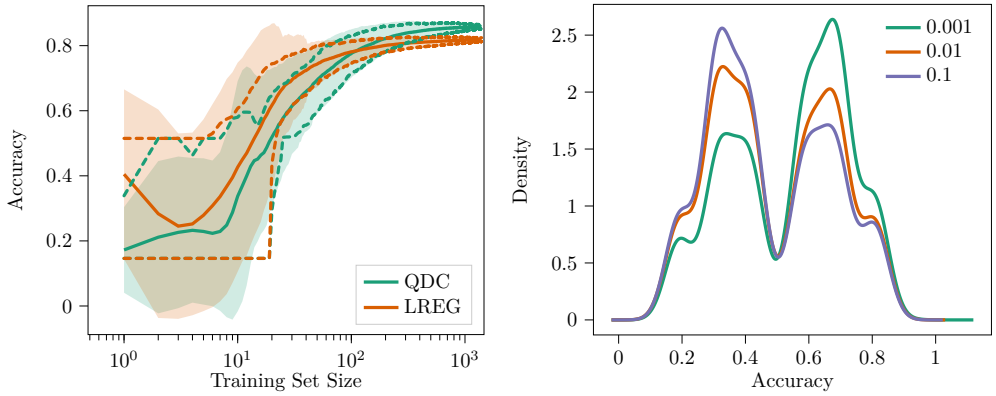
mistic estimate than the mean. In finance, the Value-at-Risk measure is widely used to quantify potential losses at a specified probability level [2], and in medical applications such as drug discovery, higher quantiles can be used to evaluate promising candidates [3].

Average performance is used for obtaining learning curves, model selection and hyper-parameter tuning. Although generally non-parametric tests are available, often parametric ones are utilized without being able to test the assumptions of it. For instance, a set of performance estimates are often used with (paired) t-tests to determine if the expected performance of models are significantly different in all of these settings with limited number of samples. This prevents accurately testing if actually the individual model performance or their differences follow a Gaussian distribution or not [4]. This assumption is often justified by viewing each prediction as a Bernoulli trial, leading to a Binomial distribution over errors. For large and fixed test sets, the Binomial distribution is assumed to converge asymptotically to a Gaussian [5]. However, when we use finite datasets, this assumption may not hold.

To highlight that generalization performance along learning curves might be complex, we show an example on a OpenML-46597 classification problem in Figure 2.1a. Here, the generalization accuracy of Quadratic Discriminant classifier (QDC) and Logistic Regression classifier (LREG) without regularization are plotted against training-set size (n), with the solid lines representing the average performance, and the dotted lines representing 5% and 95% quantiles. The shaded area indicates the two standard deviations away from the average performance. If we look at average performance, QDC is preferred over LREG for a training set size around $n > 100$. When the 5% quantile is used for comparison QDC has higher accuracy for $n > 200$. However, if we choose the 90% quantile, the threshold where one chooses QDC over LREG earlier $n > 80$.

This issue does not only pertain to the learning curves, but can also affect other subsampling based generalization performance estimation strategies. Figure 2.1b shows the accuracy distribution of a multi-layer perceptron (5 hidden layers with 16 neurons) resulting from 20-fold cross-validation repeated 1000 times on a OpenML-1046 classification task. We can clearly observe a bimodal distribution for generalization performance, one below 0.5 accuracy level and one above. This means that some models resulting from the cross-validation procedure perform worse than random chance and some not. Across different learning rates, the dominant mode of the performance distribution can change. When conducting cross-validation with a limited computational budget, one may miss some of these modes and end up selecting a suboptimal learning rate.

In this paper, we systematically analyze the distributions of generalization per-



(a) Learning curves of Logistic Regression and Quadratic Discriminant models for OpenML-46597.Experiments are repeated 1000 times

(b) Multi-layer Perceptron with 20-fold cross-validation repeated 1000 times on OpenML-1046 for various learning rates.

Figure 2.1: Performance Distributions for learning curve and cross-validation.

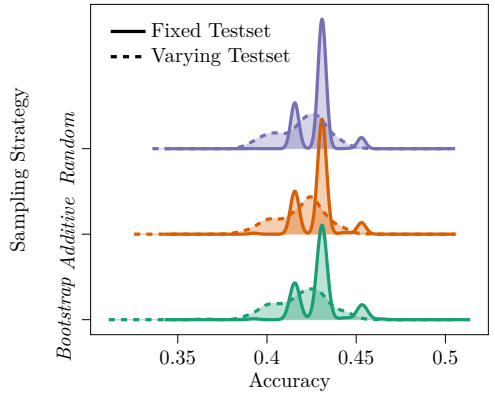
formance obtained for learning curves across a wide range of models, datasets, sampling strategies, and evaluation metrics. To this end, we create a high-fidelity learning curve database (LCDB++). We refer to it as high-fidelity since the biases introduced by predefined sampling grids or the limited number of repetitions that affect other databases [6, 7] are mitigated. We examine the extent to which the common Gaussianity assumption holds for this database and explore how factors such as dataset balance, model type, loss functions, multi-class extensions, hyper-parameter tuning, and sampling strategies influence this. Finally, we assess the practical impact of these deviations by comparing model selection outcomes when using traditional statistics (mean and standard deviation) versus alternative measures such as quantiles.

2.2 Experimental Setup

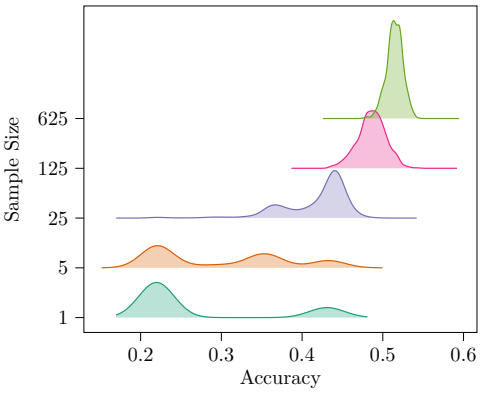
We investigate the generalization performance along learning curves, which show generalization performance with respect to training set size. In this section, we formalize our definition of a learning curve and introduce how we constructed our learning curve database.

2.2.1 Generalization Performance Along Learning Curves

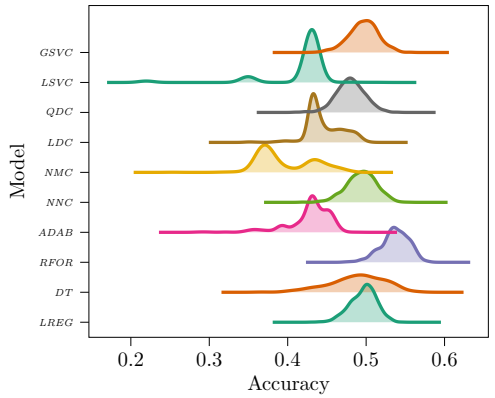
Let us denote the input and output spaces as $\mathbb{X}$ and $\mathbb{Y}$, respectively. A learning algorithm $\mathcal{A}$ takes a training set $\mathcal{D}_n := (x_i, y_i)_{i=1}^n$, which is sampled from a dataset containing i.i.d. samples $\mathcal{D}_N := (x_i, y_i)_{i=1}^N$ from unknown distribution $\mathcal{P}(x, y)$ over $\mathbb{X} \times \mathbb{Y}$. Providing this algorithm with training data result in a hypothesis h from a class $\mathbb{H}$: $h_n := \mathcal{A}(\mathcal{D}_n) \in \mathbb{H}$. Note that, producing a hypothesis can involve hyper-parameter



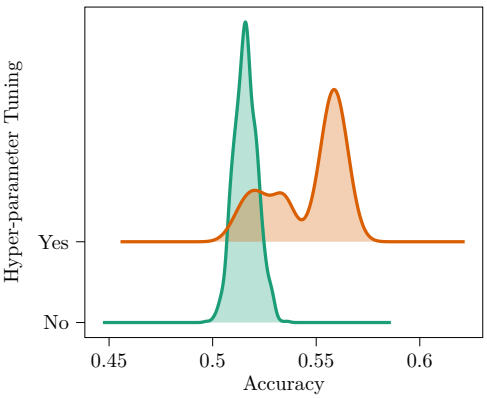
(a) Different sampling strategies at sample size 500 for ADAB.



(b) Different sample sizes for QDC $n = (1, 5, 25, 125, 625)$.



(c) Different models at sample size 100.



(d) Hyper-parameter tuning QDC at sample size 900.

Figure 2.2: Investigating the performance distribution in various settings for OpenML-1063 dataset with fixed testset.

tuning as well. Then, the prediction of a learner can be represented as $\hat{y} = h_n(x) \in \mathbb{Y}$. The performance of the learner is measured by a loss function $\mathcal{L}(y, \hat{y})$. The expected performance $\mathcal{R}$ of a hypothesis h over the true distribution $\mathcal{P}(x, y)$ is given by:

$$\mathcal{R}(h_n) = \int \mathcal{L}(y, \hat{y}) \mathcal{P}(x, y) dx dy \tag{2.1}$$

The true performance in Equation 2.1 is attainable only when we have access to $\mathcal{P}(x, y)$ and when the integral is tractable. In practice, the risk is usually estimated through cross-validation, bootstrapping, or other related methods [8]. A learning curve can then be obtained by increasing the training set size n, repeatedly sampling

training sets $\mathcal{D}_n$ from a finite dataset $\mathcal{D}_N$, and computing the average risk

$$\bar{\mathcal{R}}_n = \mathop{\mathbb{E}}_{\mathcal{D}_n \sim \mathcal{D}_N} \mathcal{R}(\mathcal{A}(\mathcal{D}_n)). \tag{2.2}$$

2.2.2 High-fidelity Learning Curve Database

While obtaining learning curves, two main decisions arise. The first is the choice of training set sizes used to evaluate generalization performance, and the second is the number of repetitions performed for each size. To obtain reliable distributions of performance, experiments must be repeated many times, and to study these distributions along learning curves, we use a finely spaced grid of training set sizes. We construct generalization performance distributions along learning curves for every training set size $p_{\mathcal{R}_n}$. This enables us to go beyond average performance and investigate additional statistical measures, such as quantiles $F^{-1}_{\mathcal{R}_n}(q)$, which are defined via the cumulative distribution function (c.d.f.) of the performance $F_{\mathcal{R}_n}(r) := p(\mathcal{R}_n \leq r) = q$.

The database contains the generalization performance of Logistic Regression (LREG), Linear and Gaussian Kernel Support Vector Machine (LSVC, GSVC), Linear Discriminant and Quadratic Discriminant (LDC, QDC), Nearest Mean (NMC), Decision Tree (DT), Random Forest (RFOR), AdaBoost (ADAB) classifiers. Since some of these methods do not support multi-class classification we use one-vs-rest for generalization performance estimates. In our experiments we investigated 10 datasets from OpenML database. We utilize the four widely used performance measures for classification tasks. Area Under the ROC curve (AUC), Brier Loss (BRI), Accuracy (ACC) and Cross-Entropy Loss (CE) . Note that AUC is only for binary classification, hence we use one-vs-rest strategy to generalize two-class classifiers to multi-class classifiers [9]. Every training set size in the range $n \in [1, 0.7 * N]$ for fixed models and $n \in [10, 0.7 * N]$ for hyper-parameter tuned learning curves is used. Moreover, for every training set size we repeat the experiments 1000 times.

To investigate the effect of splitting and sampling schemes used in learning curve generation, we also investigate most common schemes. One of the ways of obtaining a learning curve is using all the data and the other is separating a test set at the beginning [8]. If we choose to use the whole dataset as the training samples increase the test set size will decrease, hence we refer this choice as *Varying Testset*. For the case where the dataset is split at the start is referred as *Fixed Testset*. For obtaining multiple training datasets for a given training set size one may employ many sampling strategies, such as input density preserving sampling, cross-validation, random sampling, bootstrap method etc [10]. We choose the simplest settings to investigate the effect of sample replacement and sample dependence. Hence, we use *Random* sampling

as randomly drawing samples without replacement, *Bootstrap* sampling with replacement, and finally *Additive* involves drawing random points without replacement and incrementally adding them to the training dataset.

To investigate the effect of splitting and sampling schemes in learning curve generation, we consider the most common approaches. One way to obtain a learning curve is by using all available data, while another is by separating a test set at the beginning [8]. If the whole dataset is used, the size of the test set decreases as the number of training samples increases. We refer to this approach as *Varying Testset*. In contrast, when the dataset is split at the start, we call it *Fixed Testset*. To obtain multiple training datasets for a given training size, several sampling strategies may be employed, such as input density preserving sampling, cross-validation, random sampling, or the bootstrap method [10]. In our study, we focus on the simplest settings to examine the effect of sample replacement and sample dependence. Specifically, we use *Random* sampling, where samples are drawn without replacement; *Bootstrap* sampling, where samples are drawn with replacement; and *Additive* sampling, where random samples are drawn without replacement and incrementally added to the training dataset.

A visual summary of the possible samplings is shown in Figure A.1, and information about the OpenML datasets used can be found in A.1. Finally, our database can be found at (LCDB++).

2.3 Results

In this section, we investigate if the generalization performance distributions follow a Gaussian distribution and the effect of it. This investigation is conducted from various aspects, including model family, performance metric, and sampling strategy. Samples of generalization performance distributions from our learning curve database are presented in Figure 2.2, along with density estimations of the observed generalization performance for several cases. Results in this section pertain only to Accuracy and AUC. Furthermore, unless pairwise comparisons are not mentioned the whole

Table 2.1: Percentage of non-normal performance distributions for each loss measure and across dataset indicated by their OpenML IDs.

OpenML-ID	**11**	**23**	**37**	**53**	**61**
Accuracy	98.16 ± 6.28 %	99.75 ± 0.46 %	97.63 ± 6.68 %	94.85 ± 13.75%	99.38 ± 2.47 %
AUC	92.27 ± 12.85%	97.14 ± 5.96 %	93.57 ± 21.84%	82.15 ± 28.38%	91.51 ± 13.61%
OpenML-ID	**1063**	**44037**	**46597**	**46733**	**46847**
Accuracy	93.77 ± 9.98 %	91.95 ± 16.79%	88.10 ± 19.30%	97.76 ± 9.73 %	94.66 ± 15.14%
AUC	88.82 ± 15.25%	87.70 ± 21.64%	86.47 ± 16.13%	97.54 ± 4.82 %	86.63 ± 23.61%

database is used, for the paired investigations 3 missing dataset results are excluded. The corresponding results for other loss measures are presented in A.3.

The effect of sampling strategies is visualized in Figure 2.2a, using a fixed learner (QDC) and dataset (OpenML-1063). While the overall variation between sampling methods are not substantial, it is apparent that the fixing the test set size can shift the current modes and add other modes to the generalization performance. Since we do not see a significant difference between the sampling strategies, we use the *Additive* sampling strategy in the rest of the Figure 2.2 with the *Fixed Testset* as that is the mostly used setting in supervised machine learning applications.

Figure 2.2b shows slices of training set sizes versus generalization performance for a fixed QDC model, and the accuracy performance metric. As seen in the figure, when the training set size is small ($n = 1$) performance distribution has two distinct modes. As the sample size increases, the distribution of generalization performance changes significantly. So, the generalization performance distribution changes with respect to training set size.

For the same dataset and training set size, different models have completely different performance distributions. In Figure 2.2c, we plot performance distributions for a fixed sample size ($n = 100$) and a fixed dataset across all models. It is evident that not only does the average of the performance distributions vary, but the shapes of the distributions also differ. LSVC displays a heavy-tailed performance distribution, NMC has bi-modal distribution, while DT has a more uniform spread. In contrast, the ADAB exhibits a multiple modes that are close to each other.

Tuning the hyper-parameters for the same sample size of one model (QDC in this case) can also alter the generalization performance distribution compared to fixed model. As shown in Figure 2.2d, although the average of the performance improves with tuning, it creates another mode that is close to the not tuned version performance, indicating that tuning can still lead to poorly performing models in some cases.

As illustrated in Figure 2.2, the distribution of generalization performance can vary

Table 2.2: Percentage of non-normal performance distributions for each loss measure and model.

	ADAB	**DT**	**GSVC**	**LDC**	**LREG**
Accuracy	99.58 ± 2.21%	97.90 ± 5.12%	97.25 ± 8.14%	99.14 ± 2.33%	95.99 ± 8.43%
AUC	94.55 ± 5.12%	93.56 ± 7.98%	96.09 ± 10.37%	88.83 ± 14.34%	94.99 ± 9.73%
	LSVC	**NMC**	**NNC**	**QDC**	**RFOR**
Accuracy	92.87 ± 18.94%	97.91 ± 6.04%	92.79 ± 13.59%	85.64 ± 22.16%	98.07 ± 5.53%
AUC	97.28 ± 8.82%	89.70 ± 22.09%	85.07 ± 19.30%	69.43 ± 35.53%	95.58 ± 8.84%

significantly depending on how learning curves are obtained, and may deviate from Gaussianity. In the following section, we qualitatively investigate how frequently this deviation occurs using our learning curve database, derived from various real-world datasets.

2.3.1 Deviations from the Gaussian Distribution

In this section, we present results from Shapiro-Wilk tests on performance estimates for each sample size to determine whether the generalization performance distributions are Gaussian. The significance level for the all the tests is set at $\alpha = 0.05$.

In Table 2.1, we present the percentages of non-Gaussian generalization performance distributions, along with corresponding standard deviations. For each training set size, we assess whether the distribution of generalization performance is Gaussian along learning curves.

Across all loss functions and datasets, approximately 95% of the generalization performance distributions are found to be non-Gaussian. For nearly all datasets, the accuracy metric consistently shows the lowest level of Gaussianity, with only around 5% of distributions classified as Gaussian. This shows that it is rare to find performance distributions that are Gaussian.

Among all datasets, OpenML-23 exhibits the lowest proportion of Gaussian distributions, approximately 2%. This dataset is imbalanced, containing a different number of samples across all available classes. In contrast, OpenML-46597, which is balanced (with the same number of samples per class), exhibits the highest percentage of Gaussian cases. To better capture this distinction, Table 2.3 also reports our statistics separately for balanced and imbalanced datasets. Our analysis indicates that data imbalance has only a marginal overall effect for the accuracy metric. However, for *AUC* metric there is around 6% increase in Gaussian distributed performance. A similar result is observed for Brier Loss, see A.5. This indicates that performance metrics can be affected by data imbalance.

Table 2.3: Effect of dataset imbalance to the Gaussianity.

	Imbalanced	**Balanced**
Accuracy	96.95%	93.57%
AUC	92.66%	86.96%

Different models lead to different performance distributions An example of this is shown in Figure 2.2c. To assess whether certain models are more prone to producing non-Gaussian generalization performance distributions, Table 2.2 reports the

percentage of non-Gaussian cases observed for each model across different loss functions. Our selection covers a diverse range of model families, including ensemble methods, gradient-based learners, and models with analytical solutions. Among all models, ADAB shows the lowest propensity for Gaussian generalization performance, with approximately 1% of its accuracy distributions classified as Gaussian. This effect is slightly reduced when using the AUC metric. By contrast, the QDC model yields 30% Gaussian distributions for the AUC.

There are several modeling decisions involved in solving supervised classification problems. One key decision arises when adapting a binary classifier to handle multi-class classification tasks. In our setup, we use the one-vs-rest strategy to enable this extension. To evaluate the effect of this approach, we focus on the Logistic Regression model applied to both binary and multi-class problems, as shown in Table 2.4. Our results indicate that this modeling has different effects for performance metrics. The accuracy metric shows minimal sensitivity to the transition from binary to multi-class classification with a slight increase in normal distributions. In contrast, the AUC metric exhibits a decrease. This is expected since it also requires adaptation through the one-vs-rest scheme in multi-class settings.

Table 2.4: Percentage of non-normal and loss measure for binary and multi-class cases of LREG.

	Binary	**Multi-class**
Accuracy	96.99%	94.78%
AUC	93.21%	96.43%

Hyper-parameter tuning increases the non-Gaussian performance distributions In Table 2.5, we examine the effects of hyperparameter tuning. Four different models are considered are, DT, LDC, LREG, and RFOR, which represent a diverse set of modeling approaches. In our database learning curves for hyper-parameter-tuned models are shorter than their untuned counterparts since their starting point is training set size $n = 10$ instead of $n = 1$. Hence, we restrict the analysis of untuned models to the same sample sizes used in the tuned setting. The evaluation metric is accuracy for consistency. For all selected models, hyper-parameter tuning results in slight increase for non-Gaussianity of the generalization performance. In the case of the RFOR model, performance is entirely non-Gaussian.

Normality of paired performance differences Table 2.6 reports the proportion of non-normal performance difference distributions across all model combinations. We

Table 2.5: Percentage of non-normal test splits for each dataset for Accuracy and a fixed testset.

	DT	*LDC*	*LREG*	*RFOR*
Not Tuned	97.67±6.03%	99.80± 0.57 %	97.89 ±5.47%	97.82±5.47%
Tuned	98.73±4.84%	99.50 ± 1.67%	98.72 ±4.35%	100.0 ±0.00%

observe that when fixed models are evaluated on varying test sets, the paired differences between models are more likely to follow Gaussian distributions, around 20% of the cases, compared to learning curves obtained from tuned models on fixed test sets, where this holds in only about 1% of the cases. Notably, neither model tuning alone nor the choice between fixed and varying test sets shows such a pronounced increase in the normality of paired performance difference distributions.

Table 2.6: Percentage of non-normal paired generalization performance differences for all model combinations.

	Accuracy	**AUC**
Not Tuned + Varying Testset	84.85%	81.25%
Tuned + Fixed Testset	99.24%	98.40%

Smaller training set sizes exhibit less normally distributed generalization performance. Thus far, we have established that generalization performance distributions along learning curves are predominantly non-Gaussian, regardless of the assumptions made during the construction of the curves. To illustrate the spatial dependence of this phenomenon with respect to sample size, we normalize all learning curves obtained, with Additive sampling and Fixed Testset, to the range $[0, 1]$ with respect to the training set size and report the frequency of observed non-Gaussian distributions across this normalized domain. As depicted in Figure 2.3, the general trend indicates that the likelihood of observing Gaussian generalization distributions increases with larger training sample sizes. On our preliminary inspection we found an exception for the AUC metric compared to others, where the initial segments appear to exhibit more Gaussian behavior. This, however, is largely due to the lack of variation, where the metric often remains constant at 0.5. In such cases, the Shapiro–Wilk test loses validity due to zero variance.

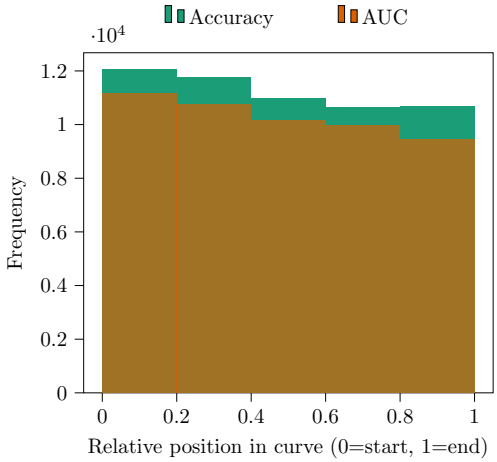


Figure 2.3: The frequency of performance distributions that are Gaussian, as a function of the training set size.

2.3.2 Gaussianity Assumption Gone Wrong

In the previous section, we demonstrated that the assumption of Gaussianity is frequently violated along learning curves. In this section, we investigate the consequences of this violation. Specifically, we analyze the learning curves of all models across all datasets for the *Additive* sampling strategy with Fixed Testset and examine whether the rankings of the top-3 models change.

Our baseline is the commonly used approach of selecting models based on the mean and standard deviation, which implicitly assumes Gaussian performance distributions. We compare this with selection based on alternative statistical measures: the 0.975-quantile, the median (0.5-quantile), and the 0.025-quantile. To ensure a consistent comparison between quantiles and the mean, we evaluate the 0.975/0.025-quantiles relative to the mean using the interval $\mu \pm 2\sigma$, where μ and σ are the mean and standard deviation of the performance distribution, respectively.

The results are summarized in Table 2.7. From the table, we observe that selection based on the mean often resembles selection based on lower quantiles, with the probability of a different model ordering around 0.40 for accuracy and 0.44 for AUC. Comparing the mean with the median shows a slightly higher probability of reordering, while the 0.975-quantile produces the most conflicts, with a 0.94 probability that the top-3 models differ in ranking or composition. This indicates that model ordering changes significantly across different quantiles depending on the performance metric, and that the top models change considerably when different statistical measures are used, especially when the best-performing models are of interest.

Table 2.7: Top-3 model Probability of observing a change in the top 3 models for Accuracy and AUC Measures with Additive Sampling where the test set is separate and hyper-parameter tuning is done. Excluding the 3 datasets that had missing learning curves.

	Accuracy	**AUC**
$\mu + 2\sigma$ *vs 0.975-Quantile*	0.94	0.90
μ *vs Median*	0.42	0.44
$\mu - 2\sigma$ *vs 0.025-Quantile*	0.40	0.44

2.4 Discussion and Conclusion

In this work, we investigate the distribution of classifier performance across multiple datasets, evaluation measures, sampling strategies, and training set sizes. We find that, in most cases, performance distributions deviate from a Gaussian distribution. In particular, smaller training sets often produce distributions with long tails and multiple modes, and hyper-parameter tuning amplifies the non-Gaussian behavior. This effect is consistent across different test splits and sampling strategies, although the choice of sampling method itself has only a minor impact on Gaussianity.

The results reported in [11, 12] suggest that the normality assumption is well justified for many sources of variation in hyper-parameter optimization for deep learning applications, which are not covered in our study. Nevertheless, not all generalization performances in these studies conform to a Gaussian distribution also. Moreover, their experimental setup differs from ours: their sources of variation do not include training set sizes. Our spatial analysis along learning curves further reveals that smaller training sets are more likely to yield non-Gaussian distributions, whereas larger sets tend to produce distributions closer to Gaussian behavior, partially explaining the patterns observed in prior work. High-fidelity learning curve generation for deep learning models is needed to fully verify these results.

We also show the deviations from Gaussianity have practical consequences for model selection. When comparing models using the mean and standard deviation versus robust statistics such as quantiles and medians, substantial discrepancies arise. For instance, ranking models by the 0.95 quantile instead of the mean changes the top three models in 90% of the cases, highlighting that relying solely on mean performance can be misleading.

Average learning curves can be used for model selection [13] faster and equally reliable as cross-validation. We argue that quantile-based learning curves can provide additional insights into training set size requirements and model robustness. For ex-

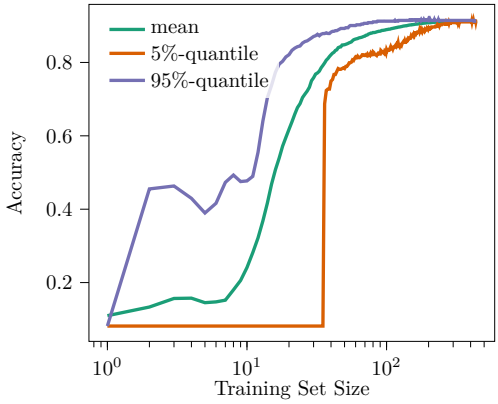


Figure 2.4: Quantile learning curves for the Quadratic Discriminant Classifier on the OpenML-11 dataset.

ample, in Figure 2.4, the average performance of the Quadratic Discriminant Classifier on the OpenML-11 dataset increases steadily with more training data, yet the lower quantile remains flat, indicating that poorly performing runs do not improve until the training set size reaches $n = 40$. Conversely, while the average performance appears to plateau around 100 samples, the lower quantile begins to improve, revealing delayed benefits for underperforming runs. These patterns demonstrate that averages can obscure critical information about reliability and robustness.

The failure of the Gaussianity assumption has further implications. In particular, it violates assumptions underlying (paired) t-tests commonly used for algorithms trained with random sub-sampling. Although we did not use the exact sampling strategies from [14, 15, 16], similar assumptions may be violated there as well, given that we observed no significant differences between sampling methods.

For more reliable comparisons, we recommend evaluating quantiles of generalization performance distributions using the statistical frameworks presented in [17, 18] or investigating stochastic dominance as proposed in [19]. While generating well-sampled performance distributions for accurate quantile estimation is computationally demanding [20], such rigor is essential for benchmarking and model comparison to fulfill their intended scientific purpose [21, 22], particularly in data-scarce or high-uncertainty settings.

References

[1] Christian Fröhlich and Robert C. Williamson. *Tailoring to the Tails: Risk Measures for Fine-Grained Tail Sensitivity*. 2023. arXiv: 2208.03066 [cs.LG].

[2] Zhijie Xiao, Hongtao Guo, and Miranda S. Lam. "Quantile Regression and Value at Risk". In: *Handbook of Financial Econometrics and Statistics*. Ed. by Cheng-Few Lee and John C. Lee. New York, NY: Springer New York, 2015, pp. 1143–1167. DOI: 10.1007/978-1-4614-7750-1_41.

[3] O Watson et al. "A Decision-Theoretic Approach to the Evaluation of Machine Learning Algorithms in Computational Drug Discovery". In: *Bioinformatics (Oxford, England)* 35.22 (2019), pp. 4656–4663.

[4] Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge: Cambridge University Press, 2011. DOI: 10.1017/CBO9780511921803.

[5] John Langford. "Tutorial on Practical Prediction Theory for Classification". In: *Journal of Machine Learning Research* 6.10 (2005), pp. 273–306.

[6] Felix Mohr et al. "LCDB 1.0: An Extensive Learning Curves Database for Classification Tasks". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Massih-Reza Amini et al. Vol. 13717. Cham: Springer Nature Switzerland, 2023, pp. 3–19. DOI: 10.1007/978-3-031-26419-1_1.

[7] Cheng Yan, Felix Mohr, and Tom Viering. *LCDB 1.1: A Database Illustrating Learning Curves Are More Ill-Behaved Than Previously Thought*. 2025. arXiv: 2505.15657 [cs.LG].

[8] Tom Viering and Marco Loog. *The Shape of Learning Curves: A Review*. Nov. 2022. arXiv: 2103.10948 [cs].

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. New York: Springer, 2006.

[10] Marcin Budka and Bogdan Gabrys. "Correntropy-Based Density-Preserving Data Sampling as an Alternative to Standard Cross-Validation". In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. Barcelona, Spain: IEEE, July 2010, pp. 1–8. DOI: 10.1109/IJCNN.2010.5596717.

[11] Xavier Bouthillier et al. "Accounting for Variance in Machine Learning Benchmarks". In: *Proceedings of Machine Learning and Systems*. Ed. by A. Smola, A. Dimakis, and I. Stoica. Vol. 3. 2021, pp. 747–769.

[12] Christoph Lehmann and Yahor Paromau. *Quantifying Uncertainty and Variability in Machine Learning: Confidence Intervals for Quantiles in Performance Metric Distributions*. Jan. 2025. DOI: 10.48550/arXiv.2501.16931. arXiv: 2501.16931 [cs].

[13] Felix Mohr and Jan N. van Rijn. "Fast and Informative Model Selection using Learning Curve Cross-Validation". In: *CoRR* abs/2111.13914 (2021). arXiv: 2111.13914.

[14] Thomas G. Dietterich. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms". In: *Neural Computation* 10.7 (Oct. 1998), pp. 1895–1923. DOI: 10.1162/089976698300017197.

[15] Remco R. Bouckaert and Eibe Frank. "Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Takeo Kanade et al. Vol. 3056. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 3–12. DOI: 10.1007/978-3-540-24775-3_3.

[16] Janez Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *J. Mach. Learn. Res.* 7 (Dec. 2006), pp. 1–30.

[17] Xinmin Li et al. "Comparison of Quantiles for Several Normal Populations". In: *Computational Statistics & Data Analysis* 56.6 (June 2012), pp. 2129–2138. DOI: 10.1016/j.csda.2012.01.002.

[18] Rand R. Wilcox et al. "Comparing Two Independent Groups via the Lower and Upper Quantiles". In: *Journal of Statistical Computation and Simulation* 84.7 (July 2014), pp. 1543–1551. DOI: 10.1080/00949655.2012.754026.

[19] Rotem Dror, Segev Shlomov, and Roi Reichart. "Deep Dominance - How to Properly Compare Deep Neural Models". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen, David Traum, and Lluís Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 2773–2785. DOI: 10.18653/v1/P19-1266.

[20] E. Niclas Jonsson and Joakim Nyberg. "A Quantitative Approach to the Choice of Number of Samples for Percentile Estimation in Bootstrap and Visual Predictive Check Analyses". In: *CPT: Pharmacometrics & Systems Pharmacology* 11.6 (Apr. 2022), p. 673. DOI: 10.1002/psp4.12790.

[21] Mostafa Dehghani et al. *The Benchmark Lottery*. 2021. arXiv: 2107.07002 [cs.LG].

[22] Odd Erik Gundersen et al. “On Reporting Robust and Trustworthy Conclusions from Model Comparison Studies Involving Neural Networks and Randomness”. In: *Proceedings of the 2023 ACM Conference on Reproducibility and Replicability*. ACM REP ’23. New York, NY, USA: Association for Computing Machinery, June 2023, pp. 37–61. DOI: 10.1145/3589806.3600044.

3 | Learning Learning Curves

Learning curves depict how a model's expected performance changes with varying training set sizes, unlike training curves, showing a gradient based model's performance with respect to training epochs. Extrapolating learning curves can be useful for determining the performance gain with additional data. Parametric functions, that assume monotone behaviour of the curves, are a prevalent methodology to model and extrapolate learning curves. However, learning curves do not necessarily follow a specific parametric shape: they can have peaks, dips, and zigzag patterns. These unconventional shapes can hinder the extrapolation performance of commonly used parametric curve-fitting models. In addition, the objective functions for fitting such parametric models are non-convex, making them initialization-dependent and brittle. In response to these challenges, we propose a convex, data-driven approach that extracts information from available learning curves to guide the extrapolation of another targeted learning curve. Our method achieves this through using a learning curve database. Using the initial segment of the observed curve, we determine a group of similar curves from the database and reduce the dimensionality via Functional Principle Component Analysis FPCA*. These principal components are used in a semi-parametric kernel ridge regression (*SPKR*) model to extrapolate targeted curves. The solution of the* SPKR *can be obtained analytically and does not suffer from initialization issues. To evaluate our method, we create a new database of diverse learning curves that do not always adhere to typical parametric shapes. Our method performs better than parametric non-parametric learning curve-fitting methods on this database for the learning curve extrapolation task.*

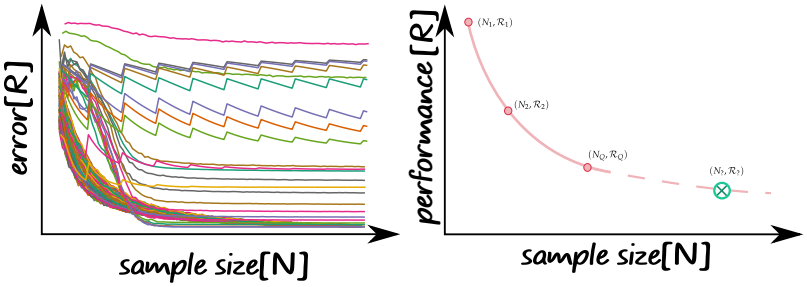
3.1 Introduction

A learning curve shows the generalization performance of a learner as a function of the training set size. This should not be confused with training curves. Both curves find various applications in machine learning pipelines. Tuning hyperparameters, selecting models, and assessing whether adding more data benefits a learner, for instance, are some of the applications [1].However, training curves are used to track model performance for a single learning problem during loss optimization, whereas

Published as: **O. Taylan Turan** et al. "Learning Learning Curves". In: Pattern Analysis and Applications 28.15 (2025).

the learning curve tracks the average model performance over different learning problems from the same dataset. In other words, one obtains the training curve at no cost while training the model. However, obtaining learning curves require training the model multiple times with different subsets of the dataset, making it computationally taxing to obtain.

The performance of a learner for increasing training set sizes can be predicted by extrapolating the learning curves. Starting with very small training set sizes, the initial segment of a learning curve can be obtained in a cheap and quick manner. With this partially observed learning curve, one can extrapolate it to predict a model's performance for the given dataset without the need for excessive amount of data or computational resources.



(a) Learning Curve Database (b) The Learning Curve to extrapolate.

Figure 3.1: Learning curve plots show generalization performance vs sample size. In b: initial points observed marked with pink, and the desired training set size marked with green for which we would like to get the generalization performance.

Historically, learning curves are extrapolated by fitting to parametric functions, such as exponential or power law functions [2, 1]. These parametric models often (implicitly) assume that the learning curves are well-behaved, i.e. that increasing dataset size the generalization performance gets better. As pointed out in the survey [3], this assumption can be violated in various ways. Moreover, due to limited number of training samples available, the fitting of nonlinear functions becomes fragile and highly dependent on the initialization.

It is worth mentioning that these parametric functions are used not only to model learning curves but also for modeling training curves. One example of this usage is presented in [4], where a collection of parametric curve models is used as a prior to extrapolate training curves with a transformer. Moreover, in [5], a hand-crafted collection of parametric models is used to model learning and training curves with multiple infliction points. Finally, [6] uses trainable parametric models for the mean function of a Gaussian Process.

Using parametric models is a logical choice, if curves have underlying parametric forms. However, curves do not follow fixed parametric shapes. For this reason, obtaining these functional forms from data is a promising path to automate learning curve extrapolation further.

To break free from limitations of parametric functions, we assume the existence of a database of learning curves, as shown in Figure 3.1a. The goal is to extrapolate an unknown targeted learning curve to get a generalization performance at a specific training set size, as depicted in Figure 3.1b, without using any parametric models.

A similar setting is considered in [7], where, a non-parametric, data-driven pipeline is introduced that uses a similar setting where a learning curve database is available. However, their task is to perform a binary model selection for classification problems at unseen sample sizes. We propose a pipeline using the semi-parametric kernel ridge (*SPKR*) model for extrapolating learning curves. *SPKR* can incorporate any real-valued function in addition to training points from the targeted learning curve. We obtain these functions by functional principal component analyses (*FPCA*) and the mean of the relevant part of a database to incorporate related learning curve behaviours for better extrapolation performance for learning curves.

In addition to our method, we provide a learning curve database. The motivation to create a new learning curve database is threefold. Firstly, to have a database free from missing values, something the dataset from [1] suffers from. Secondly, we want to include some significantly non-monotonic curves, for instance, sawing [8] and dipping [9] curves, to highlight the ability of a non-parametric approach to curve extrapolation. Finally, we want to have learning curves obtained from regression problems which [1] lacks again.

This paper is organized as follows: In Section 3.2, we provide a brief background on learning curves. Section 3.3 presents common methods for extrapolating learning curves, along with our proposed methodology. We then detail our experimental setup in Section 3.4. Section 3.5 presents our results, followed by the main conclusions in Section 3.6. Before giving our learning curve definition in Section 3.2, we should mention that this paper deviates from the works proposed on training curve extrapolation [10, 11, 12, 13, 14, 15, 16], simply because of the inherent differences between learning and training curves.

3.2 Background on the Learning Curves

In this section we cover the necessary background on learning curve theory following the notation of [2]. We employ the term learning curve, as defined in the broader context of the general machine learning literature, to describe generalization performance as it relates to the number of training samples. It is worth noting that the same

term is used to refer to different types of curves in various domains. This can lead to common misconceptions, especially when dealing with the artificial neural network (*ANN*) literature, where the x-axis of the learning curve often represents the number of training iterations [17].

To formalize our definition of a learning curve for supervised regression and classification problems, let us denote the input and output spaces as $\mathbb{X}$ and $\mathbb{Y}$, respectively. A learning algorithm $\mathcal{A}$ takes N i.i.d samples $\mathcal{D}_N := (x_i, y_i)_{i=1}^N$ from an unknown distribution $\mathcal{P}(x, y)$ over $\mathbb{X} \times \mathbb{Y}$ and produces a hypothesis h from a hypothesis class $\mathbb{H}$. This can also be represented by $h := \mathcal{A}(\mathcal{D}_N)$). Then, the prediction of a learner can be represented as $\hat{y} = h(x) \in \mathbb{Y}$. The error of the learner is measured by a loss function $\mathcal{L}(y, \hat{y})$. In classification this is typically the zero-one error, and in regression the mean squared error is often used. The expected loss (or risk) $\mathcal{R}$ of a hypothesis h over the true distribution $\mathcal{P}(x, y)$ is given by:

$$\mathcal{R}(h) = \int \mathcal{L}(y, \hat{y}) \mathcal{P}(x, y) dx dy \tag{3.1}$$

An individual learning curve of a learner $\mathcal{A}$ is ideally obtained by plotting $\mathcal{R}$ against N. Thus, a learning curve $\mathcal{C} : \mathbb{Z}^+ \to \mathbb{R}$ depends on $\mathcal{A}$, $\mathcal{P}$, and N.

3.3 Extrapolating Learning Curves

Learning curve extrapolation can be formulated as a learning problem too. Assume that we are given pairs $\mathcal{Z} := (N_i, \mathcal{R}_i)_{i=1}^Q$, which are sampled points from the beginning of an arbitrary learning curve. Our primary goal is to learn this unknown learning curve $\mathcal{C}$ such that, predicted risk $\hat{\mathcal{R}}_{\text{target}}$ is as close to the real risk $\mathcal{R}_{\text{target}}$ as possible for a given sample size $N_{\text{target}} \notin [N_1, N_Q]$. In the rest of this section, we first recall parametric curve-fitting and subsequently present our proposed approach for extrapolating learning curves.

3.3.1 Parametric Curve-fitting

Commonly, it is assumed that learning curves have similar shapes, making the parametric curve-fitting a commonly used approach [2]. Let us assume that an arbitrary parametric curve for fitting learning curve is represented by $f_\cdot(N, \boldsymbol{\theta})$, with $\boldsymbol{\theta}$ as the adjustable model parameters. Then, for training pairs $\mathcal{Z} := (N_i, \mathcal{R}_i)_{i=1}^Q$ the least squares curve-fitting problem is given by:

$$\hat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{Q} (\mathcal{R}_i - f_\cdot(N_i, \boldsymbol{\theta}))^2. \tag{3.2}$$

According to [2, 1], many researchers use power law and exponential parametric models for $f_\cdot$. To account for behaviour changes for different regions of the learning

curves, more complex parametric formulations are proposed in [5, 18]. All the parametric models proposed are non-linear functions of $\boldsymbol{\theta}$. This makes parametric curve-fitting initialization dependent. When this is the case, Equation 3.2 does not have a closed form solution and is non-convex.

3.3.2 Semi-parametric Kernel Ridge (*SPKR*)

Next to the training pairs $\mathcal{Z} := (N_i, \mathcal{R}_i)_{i=1}^{Q}$, we now assume that other learning curves are available at training time.

Let us assume a model in the form $\tilde{f} = f + h$ to approximate an arbitrary learning curve $\mathcal{C}$. Here, $f \in \mathbb{R}^{\mathbb{X}}$ represents the information coming from $\mathcal{Z}$, and $h \in span\{\psi_p\}$, where $\{\psi_p\}_{p=1}^{M}$ is a set of real-valued functions that represent the information coming from the available learning curves. Assuming a strictly increasing loss function $\mathcal{L}$, and a strictly increasing regularizer function Ω, our learning problem can be expressed as:

$$\hat{\tilde{f}} \in \arg\min_{\tilde{f} \in \mathbb{H}} \mathcal{L}(\tilde{f}, \boldsymbol{\mathcal{R}}) + \Omega(||f||_{\mathbb{H}}). \tag{3.3}$$

In this equation, $\mathbb{H}$ is the Reproducing Hilbert Space and $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{Q \times 1}$ is all the labels of our training set. According to the *Semi-parametric Representer Theorem* [19], the solution to Equation 3.3 has the form: $\tilde{f}(\cdot) = \sum_i^Q \alpha_i K(\cdot, N_i) + \sum_j^M \beta_j \psi_j(\cdot)$, where K is any Mercer kernel. If we assume $\Omega(||f||_{\mathbb{H}}) := \lambda ||f||_{\mathbb{H}}^2$ and the squared error as loss function $\mathcal{L} = \sum_{i=1}^{Q} (\mathcal{R}_i - \hat{f}(N_i, \boldsymbol{\theta}))^2$, the convex optimization problem has a unique solution. The optimal parameters are represented by $\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{Q \times 1}$, and $\hat{\boldsymbol{\beta}} \in \mathbb{R}^{M \times 1}$. Furthermore, when $\mathbf{K} \in \mathbb{R}^{Q \times Q}$, $\boldsymbol{\psi} \in \mathbb{R}^{Q \times M}$ are the kernel matrix, and the additional information coming from the available learning curves, respectively, this unique solution can be obtained as:

$$\hat{\mathbf{w}} = (\mathbf{A}^{\mathsf{T}}\mathbf{A} + \lambda \mathbf{B}^{\mathsf{T}})^{-1} \mathbf{A}^{\mathsf{T}} \boldsymbol{\mathcal{R}}. \tag{3.4}$$

Here, $\hat{\mathbf{w}} := [\hat{\boldsymbol{\alpha}}; \hat{\boldsymbol{\beta}}] \in \mathbb{R}^{(Q+M) \times 1}$ is the collection of the optimal parameters, and $\mathbf{A} := [\mathbf{K}, \boldsymbol{\psi}] \in \mathbb{R}^{Q \times (Q+M)}$ is the concatenation of the Kernel matrix and the additional information coming from the available learning curves, and $\mathbf{B} := [\mathbf{I}, \mathbf{0}; \mathbf{0}, \mathbf{0}] \in \mathbb{R}^{(Q+M) \times (Q+M)}$ is the regularization applied to f.

A crucial part of *SPKR* is how the $\{\psi_p\}_{p=1}^{M}$ are obtained. In our aforementioned extrapolation problem, we aim to obtain it from the relevant part of a learning curve database. First, the relevant part of the database can be selected by using a similarity measure $\mathcal{S}(\mathcal{Z}, \mathbb{C})$. It measures the similarity of $\mathcal{Z}$ with respect to the initial part of all the curves available in the learning curve database $\mathbb{C}$.

One of the most obvious choices is to use all the similar curves in the database, but note that there is an inverse operation in Equation 3.4 with the complexity $\mathcal{O}((Q + M)^3)$. This results in a computational bottleneck when the relevant part of the database

is large. To keep the computations feasible, a small set of curves representing the biggest modes of variations is derived using principal component analysis (*PCA*) [20]. Next to reducing the computational cost, it also prevents overfitting to the learning curve database. Our learning curve database consists of functionals; hence we use functional principal component analysis (*FPCA*) to extract the most important modes of variations in the learning curve database.

Using the notation of [21], the covariance of learning curve database is given by $v(s,t) = U^{-1}\sum_{i=1}^{U}(\mathcal{C}_i(s) - \mu_{\mathcal{C}}(s))(\mathcal{C}_i(t) - \mu_{\mathcal{C}}(t))$, where U is the number of learning curves present in the database and $\mu_{\mathcal{C}}$ is the mean function for the relevant part of the database. We can formulate the eigenvalue problem for the *FPCA* as follows (see also [21]):

$$\int v(s,t)\xi(t)dt = \rho\xi(s). \tag{3.5}$$

This eigenvalue problem is satisfied for each eigenfunction ξ with the corresponding eigenvalue ρ. Eigenfunctions, in this case, represent the modes of variation in the database. We choose the first M of these eigenfunctions with the largest eigenvalues ($\{\psi_p\}_{p=1}^{M} = \{\xi_p\}_{p=1}^{M}$) to be used in the *SPKR*. Since the principal components describe the principal variations around the mean $\mu_{\mathcal{C}}$ of the relevant part of the database, we also separately include it in our model definition. Thus, our final proposed model for learning curve extrapolation takes the form $\tilde{f} = f + h + \mu_{\mathcal{C}}$. Similar to Equation 3.4, the optimal solution with the addition of the database mean at the training points $\bar{\boldsymbol{\mathcal{C}}} := \{\mu_{\mathcal{C}}(N_i)\}_{i=1}^{Q}$ can be obtained as:

$$\hat{\mathbf{w}} = (\mathbf{A}^{\mathrm{T}}\mathbf{A} + \lambda\mathbf{B}^{\mathrm{T}})^{-1}\mathbf{A}^{\mathrm{T}}(\boldsymbol{\mathcal{R}} - \bar{\boldsymbol{\mathcal{C}}}). \tag{3.6}$$

The entire pipeline is summarized in Algorithm 1, where the steps for extrapolation using SPKR on learning curves are outlined in detail.

procedure SPKR($\mathbb{C}, \mathcal{Z}, \mathcal{S}, N_{target}, num\_curve, pca\_perc$)
  similarities $\leftarrow$ sort($\mathcal{S}(\mathbb{C}, \mathcal{Z})$) ▷ Sort similarities
  selected_curves $\leftarrow$ $\mathbb{C}$[similarities[num_curve]] ▷ Select similar curves
  $\psi \leftarrow FPCA(selected\_curves, pca\_perc)$ ▷ Obtain eigenfunctions
  $\hat{\tilde{f}}(\cdot) \leftarrow$ solve($(\mathbf{A}^{\mathrm{T}}\mathbf{A} + \lambda\mathbf{B}^{\mathrm{T}})^{-1}\mathbf{A}^{\mathrm{T}}(\boldsymbol{\mathcal{R}} - \bar{\boldsymbol{\mathcal{C}}})$) ▷ Obtain the optimal parameters
  return $\hat{\tilde{f}}(N_{target})$
end procedure

Algorithm 1: Extrapolation with SPKR on learning curves

3.4 Experimental Setup

This section provides details of the learning curve database generation, outlines the extrapolation setting that we consider, methodological choices, and the experimental setting.

3.4.1 Learning Curve Database

The true risk in Equation 3.1 can only be computed when we have access to the exact joint distribution $\mathcal{P}(x, y)$ and when the integral is tractable. In practice, however, we only have access to a finite sample drawn from $\mathcal{P}(x, y)$. To estimate a learning curve, we choose to repeatedly select a subset of size N from the available dataset, using the remaining data to compute the test error [2]. This process is repeated 100 times, and the resulting errors are averaged to get an estimate for Equation 3.1. We apply this procedure for each training set size in the range $N \in [2, 100]$ to generate one complete learning curve.

Our database is comprised of 11240 learning curves, of which 4840 are classification problems and 6400 are regression problems. This collection of learning curves involve curves that are known to exhibit non-monotonic behaviour (*i.e.* sawing-type [8] and special high dimensional Gaussian Process model learning curves [22], dipping phenomenon [9]) as well as monotone learning curves.

Classification models are selected to be Linear Discriminant (*LDA*), Quadratic Discriminant (*QDA*), Nearest Mean (*NMC*) and Nearest Neighbor (*NNC*) classifiers. Learning curves of these classifiers with range of hyperparameters are created for the Banana (*BAN*), Gaussian (*GAU*) and Dipping [9] (*RDIP-DDIP*) datasets. Different variants of the Banana and Gaussian datasets are obtained by varying the separation of the two classes. Moreover, For the Dipping dataset dimension of the problem (*DDIP*) and the radius of the outer class (*RDIP*) increased to create various datasets.

Regression learning curves are obtained for linear model (*LIN*) with a range of regularization, multi-layer perceptron (*ANN*) model with changing width of the 2 hidden layers with soft-sign activation function. Kernel Ridge model with Gaussian (*GKR*) and Laplace (*LKR*) kernel. Datasets used include linear (*ELN*), sinc (*ESC*), and sine (*ESN*) datasets with a homoscedastic noise. Variants of these datasets are created by increasing the variance of the added Gaussian noise. Moreover, a sawing dataset (*SAW*) [8] (used only for a linear model without bias term), and finally *DGP* prior dataset [22] (used only by Gaussian process model). Further details about the models and datasets used to create the learning curves, along with their respective abbreviations, can be found in Appendix B.1.

In our experiments, we treat classification and regression problems separately due to the distinct nature of their performance metrics. Specifically, classification tasks are

evaluated based on the error rate, while regression tasks are assessed using the mean squared error (MSE). To accommodate these differences, we create separate curve databases for classification and regression, ensuring that each is split independently. For both cases, we allocate 80% of the learning curves for training and use the remaining 20% for evaluating extrapolation performance.

3.4.2 Learning Curve Extrapolation

During extrapolation, we assume that the learning curve is partially observed; up to a maximum value for N. We consider only cases in which the first $Q = 10$, $Q = 25$, or $Q = 50$ points of the target learning curve are observed. Given a learning curve database and a targeted learning curve, our primary objective is to predict the performance at the end of each targeted learning curve, $N = 100$. Finally, the extrapolation error is calculated using the squared loss for each curve for every extrapolation method. All the curves are normalized such that the area under each curve is 1.

Baselines

We choose the parametric curve-fitting baselines as *WBL4* [23] and *MMF4* [24], with parametric functions given in Equations 3.7 and 3.8 respectively.

$$f_{\text{wbl4}}(N, \boldsymbol{\theta}) = -\theta_1 \exp(-\theta_2 (N)^{\theta_3}) + \theta_4 \tag{3.7}$$

$$f_{\text{mmf4}}(N, \boldsymbol{\theta}) = (\theta_1 \theta_2 + \theta_3 \cdot (N)^{\theta_4}) / (\theta_2 + (N)^{\theta_4}) \tag{3.8}$$

These two parametric functions are reported to perform the best for extrapolating learning curves in [1] for another learning curve database. We use *LBFGS* [25] to solve Equation 3.2. Since gradient approximation (*i.e.* finite difference) results are reported to be unstable for optimization, we use the exact gradients for both parametric models [14]. The optimization is repeated 100 times where the initial parameters are drawn from a normal distribution with zero mean and unit variance. For each observed curve we do a 80%-20% random train-validation split on the observed data of that curve. The best performing parameter configuration on the validation set is used to initialize the fitting procedure again with the 100% of the observed data. Note that the baseline models do not have access to the learning curve database by construction.

We also use the parametric formulation given in [5], which can handle non-monotonic curves. The training procedure in the paper is followed. A general form with k inflection points is given by:

$$f_{\text{bnsl}_k}(N, \boldsymbol{\theta}) = \theta_1 + (\theta_2 N^{-\theta_3}) \prod_{i=1}^{k} (1 + (N/\theta_{1_i})^{1/\theta_{2_i}})^{-\theta_{3_i}\theta_{2_i}}. \tag{3.9}$$

Although, authors of [5] suggest using cross-validation to obtain the number of infliction points. We train separate models up until 3 infliction points to see the general trend of these parametric models. Although 3 infliction points might not be sufficient to model some of the curves in our database, computational resources create a bottleneck given the training procedure in [5]. These baselines are tokened as *BNSLk*, where the *k* represents the number of infliction points $0, 1, 2$.

A similar method to ours is [6], where parametric models used for enhancing the extrapolation performance of Gaussian Process Regression. However, since this model depends on an expert decision regarding the saturation limit and is designed solely for classification problems, we are unable to use it as a baseline.

We compare our model also to the non-parametric method Meta-learning on Data Samples *MDS* proposed in [7]. However, since our task is to extrapolate a learning curve and approximate the generalization performance given the initial segment of the learning curve, we adjust *MDS* as follows: First, k most similar curves of the database to the target curve $\mathcal{Z}$ is selected, from learning curve database $\mathbb{C}$ with a similarity measure $\mathcal{S}(\mathcal{Z}, \mathbb{C})$. The resulting collection of curves are denoted by $\mathbb{C}_k$. These selected curves are then scaled with $s = \frac{\sum_{i=0}^{Q}(\mathcal{R}_i \mathbb{C}_k(N_i) w_i)}{\sum_{i=0}^{Q}(\mathbb{C}_k(N_i)^2 w_i)}$, where w represents an arbitrary weight for a given point. Finally, the scaled curves are averaged to predict generalization performance at N_{target} which is given by:

$$f_{\text{mds}}(N, \mathbb{C}, k) = \frac{1}{k} \sum_{i \in \mathbb{C}_k} s_i \mathcal{C}_i(N). \tag{3.10}$$

In [7] weights are defined as $w_i = N_i^2$, however [26] suggests $w_i = 2^i$ claiming improved performance. Our preliminary experiments confirmed the superiority of this weighing scheme on our database as well, hence we adopt it. Moreover, for the similarity measure we use the cosine similarity, since we did not observe significant difference between the results of average squared distance between the curves. Additionally, we set $k = 100$ to be consistent with our method. By selecting the same similarity measure and number of curves, we ensure a fair comparison between *MDS* and our proposed method.

Fitting Semi-Parametric Kernel Ridge

Our method has several hyperparameters, including the regularization parameter (λ), kernel parameters (*e.g.* length-scale of the kernel), the number of *FPCA* components, similarity measure, and set the number of curves for selecting the relevant part of the database. We choose similarity measure as cosine similarity and the number of curves to be selected from the database as 100. Then, Nadaraya-Watson smoothing [27] with default values is applied to make the eigenfunctions smoother. In this work, we select

M *FPCA* components which represent 95% of the selected subset of the database.

For our method we optimize only for the length-scale of a Gaussian Kernel length scale in the range $\gamma \in [10^0, 10^2]$ and a regularization parameter $\lambda \in [10^0, 10^1]$. Similar to the approach used for baselines, an 80%-20% train-validation split of the targeted curve is followed by a grid search for hyper-parameter optimization. Each hyper-parameter range is divided into 20 evenly spaced values in logarithmic space. The hyper-parameter configuration that yields the lowest validation error is selected for training with the full observed data of the partially observed curve. The resulting model is evaluated for extrapolation at $N = 100$. This process is carried our for each curve to be predicted.

3.5 Results and Discussion

In this section, we investigate the extrapolation performance of our model compared to parametric baselines for regression and classification learning curves. Next, we analyze the average performances of all the models along with their variances. Then we examine the partial ordering and full ordering by analyzing the empirical cumulative distribution function and average rankings. We further explore the importance of selecting the relevant part of the database through additional experiments. Finally, we compare the average ranking of our method with a non-parametric learning curve extrapolation method *MDS*.

Note that a Wilcoxon significance test showed that the error distributions of our method compared to all the parametric and non-parametric baselines is significantly different with all the *p-values* smaller than the significance level of 0.0001.

3.5.1 Extrapolation Performance

Figure 3.2 shows the extrapolation errors for the *SPKR*, and other baselines for both classification and regression learning curves. The top row shows the averaged squared error, the bottom row shows the standard deviations. The left column shows the results for the classification learning curves, while the right column the regression results.

SPKR exhibits lower average squared errors made for the targeted extrapolation point compared to baselines across all the Q values that is considered in this work. Similarly, the spread of *SPKR* errors is smaller, except $Q = 10$, where *MMF4* and *BNSL0* and *BNSL1* show lower spread. The increased spread for small training points can be caused by the lack of information selecting the relevant part of the database with little number of training points. Additionally, in our database, highly non-monotonic curves exist where the beginning of the curve is not representative of the end portion. We also observe a slight standard deviation increase for regression problems for our

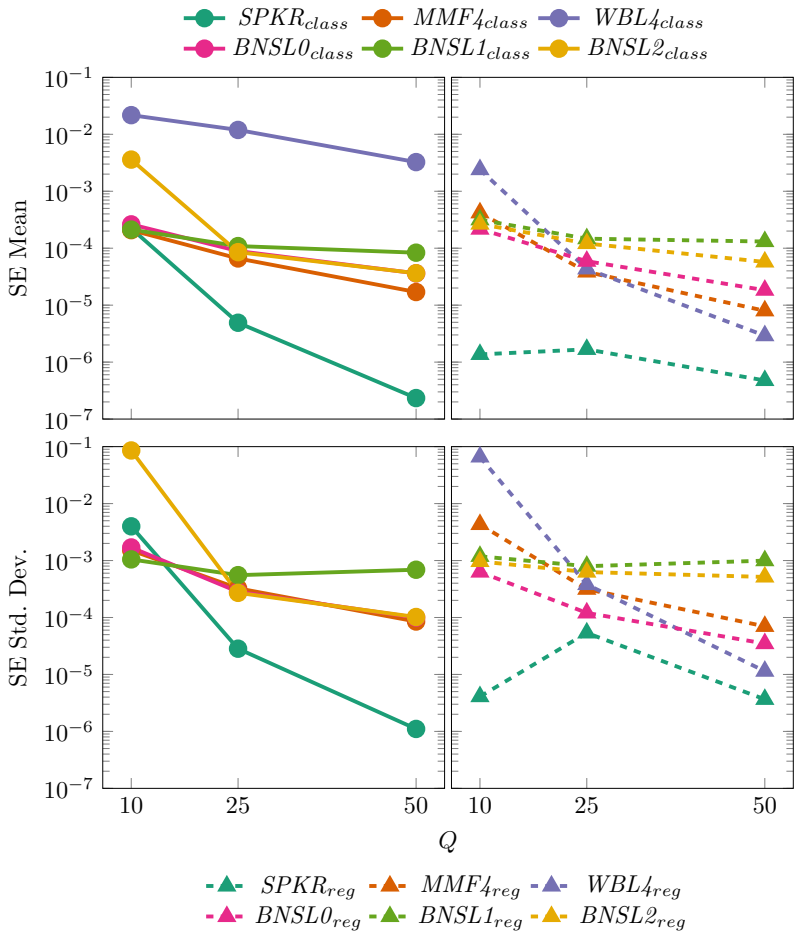


Figure 3.2: Mean and standard deviation of the extrapolation squared errors (SE) for varying training points. Classification and regression learning curve results are plotted with solid and dashed lines respectively.

method for when the training set size is increased from 10 to 25, although it still has the lowest spread.

Another observation in Figure 3.2 is that *BNSL* performance does not strictly improve as the number of infliction points are increased, hindering the trivial usage of this method in learning curve extrapolation. The varying performances of parametric models for classification and regression learning curves across varying initial segment lengths suggests that our method is able to adjust its bias dynamically with the selected curves.

Since averaging squared errors is prone to being affected by the outliers, we also

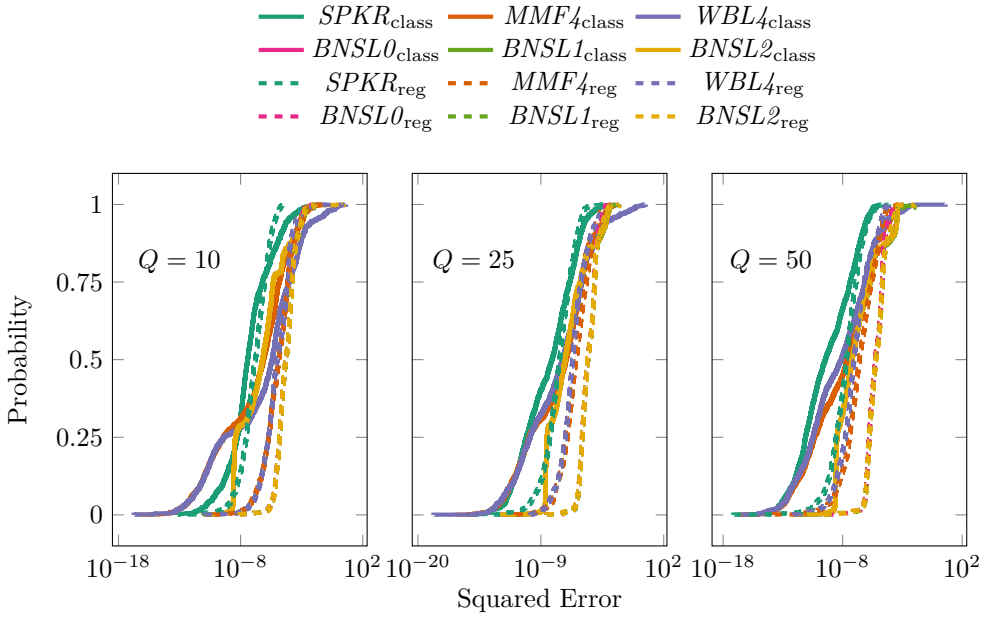


Figure 3.3: Empirical Cumulative Distribution of the extrapolation errors. Solid lines and dashed lines represent the experiments on classification and regression subsets respectively.

examine the average rankings in Table 3.1. Similar to average performance, *SPKR* achieves a better average rank for all initial segment sizes. We observe that our method is highly effective for regression problems with smaller initial segments (Q). Additionally, we investigate the average rank on subsets of the database and found that our model has a better average rank in almost all of the subsets we considered. (See Tables 3.2 and 3.3.)

Figure 3.3 illustrates the cumulative error distribution, where *SPKR* has consistently lower median for all experiments. The only instances where baselines *MMF4* and *WBL4* have lower errors is in the first quartile of $Q = 10$ for classification problems. Finally, we observe that the infliction point increase does not influence the performance of *BNSL* significantly, especially for lower Q values.

Tables 3.2 and 3.3 show the average rankings for various subsets of the data. Our method performs well across all the subsets besides the Gaussian Dataset where *MMF4* and *WBL4* has better average ranks. Only, on the Gaussian dataset (*GAU*) our method comes third.

All the results discussed so far pertain to the extrapolation performance; however, our method also performs as good in interpolation regime with the overall curve.

Table 3.1: Average extrapolation ranking (lower is better) for both classification and regression learning curves.

	Classification			**Regression**		
	$Q = 10$	$Q = 25$	$Q = 50$	$Q = 10$	$Q = 25$	$Q = 50$
SPKR	**2.5**	**2.2**	**1.8**	**1.1**	**1.3**	**1.5**
MMF4	3.4	3.2	3.0	3.1	2.7	2.6
WBL4	3.8	3.2	2.7	2.9	2.5	2.2
BNSL0	3.7	4.0	4.4	4.6	4.7	4.7
BNSL1	3.7	4.0	4.4	4.5	4.8	4.8
BNSL2	3.7	4.0	4.4	4.5	4.7	4.8

Table 3.2: Average rankings of several groupings for the classification learning curves.

	SPKR	*MMF4*	*WBL4*	*BNSL0*	*BNSL1*	*BNSL2*
NMC	**1.6**	2.8	2.4	4.7	4.6	4.6
LDC	**1.9**	2.9	2.9	4.3	4.4	4.4
QDC	**2.3**	3.5	2.8	4.1	4.0	4.0
NNC	**1.5**	2.7	2.8	4.7	4.6	4.5
DDIP	**1.8**	3.3	3.1	4.1	4.2	4.1
RDIP	**1.6**	3.2	2.9	4.4	4.3	4.3
GAU	2.2	**1.9**	1.8	4.9	4.9	4.9
BAN	**1.6**	3.4	3.1	4.3	4.2	4.2

Ranking with respect to whole curve fitting (including all extrapolation and interpolation) based on MSE can be seen in Table 3.4.

3.5.2 Obtaining ψ from database

In [28] it is argued that cosine similarity can be a problematic choice in some cases. This is why we investigated two other types of similarity measures. We found that minimizing the area between the targeted curve and the curves in the database, and dynamic time warping [29] does not yield vastly different eigenfunctions ψ in our case. Nonetheless, care must be taken when determining ψ as it the main driving force of the extrapolation performance of the *SPKR*. To demonstrate this, we intentionally choose the most dissimilar curves in our method and observe the average rank of our method drops significantly as shown in 3.5. Finally, we also attempted to get rid of the similarity measure and extract ψ via *FPCA* on the whole database, we see a similar drop in performance again for the *SPKR*.

Table 3.3: Average rankings (lower is better) of various groupings for the regression learning curves.

	SPKR	*MMF4*	*WBL4*	*BNSL0*	*BNSL1*	*BNSL2*
DGP	**1.5**	1.5	2.7	5.5	4.8	4.8
NN	**1.3**	3.1	2.7	4.4	4.6	4.6
LKR	**1.6**	2.5	1.9	4.8	4.9	4.9
GKR	**1.5**	2.6	1.9	4.9	4.9	4.9
LIN	**1.7**	2.3	2.1	4.7	5.0	4.9
ELN	**1.6**	2.6	2.1	4.7	4.8	4.8
ESC	**1.5**	2.5	2.2	4.7	4.9	4.9
ESN	**1.5**	2.6	2.2	4.7	4.8	4.8

Table 3.4: Average ranking for the whole curves (lower is better) for both classification and regression problems.

	Classification			**Regression**		
	$Q = 10$	$Q = 25$	$Q = 50$	$Q = 10$	$Q = 25$	$Q = 50$
SPKR	**1.8**	**1.7**	**1.5**	**1.3**	**1.5**	**1.6**
MMF4	2.0	2.2	2.4	2.4	2.5	2.6
WBL4	2.2	2.1	2.1	2.3	2.0	1.7

Table 3.5: Average extrapolation ranking (lower is better) for the case when we choose the least similar curves in the database.

	Classification			**Regression**		
	Q:10	Q:25	Q:50	Q:10	Q:25	Q:50
SPKR	5.8	5.7	5.8	3.9	4.8	5.7
MMF4	**2.8**	2.6	2.3	2.5	2.0	1.9
WBL4	3.2	**2.6**	**2.1**	**2.3**	**1.7**	**1.5**
BNSL0	3.0	3.3	3.6	4.1	4.0	3.8
BNSL1	2.9	3.2	3.6	4.0	4.1	3.9
BNSL2	2.9	3.2	3.5	4.0	4.0	3.9

We assume that the learning curve database contains only learning curves, with no other information available. As shown in [26], an active testing strategy proposed in [30] for learning curve selection can enable these types of curve selection strategies when the information is available, and might improve our methods extrapolation per-

formance.

3.5.3 Divergence of Parametric models

We observed diverging curve-fitting results for complex problems, which can happen for non-convex objective functions. To ensure a fairer comparison between all models, we decided to exclude all curves that were problematic for at least one model. Thus, we ended up removing 9% of our results for the test part of our learning curve database.

Since our method has analytical solution, we do not have diverging solutions. Our method is able to find the best solution that is minimizing the squared error for the training data with the obtained *FPCA* components. Moreover, although we eliminate the diverging results of the parametric models our proposed method has lower variance in most of the cases, making our model more reliable.

3.5.4 Comparison with *MDS*

Performance across classification and regression tasks are presented in Tables 3.6 and 3.7. For both types of learning curves, *SPKR* achieved the lowest rankings, outperforming *MDS* in all experiments, except for the classification learning curves with smaller observed part $Q = 10$.

The better performance of our method is expected since it incorporates the *MDS* method. Prediction of *MDS* is based solely on the average of the obtained learning curves from the database. We assumed a solution in the form $\tilde{f} = f + h + \mu_{\mathcal{C}}$, where $\mu_{\mathcal{C}}$ is the mean of the learning curves obtained from the database. Hence, on top of the mean of the most similar curves, we also leverage data points and *FPCA* components from the database to improve our prediction compared to *MDS*. In addition, our method relies on interpolation of the learning curves in the database as it is required for the *FPCA*. This makes our method more robust for cases where the learning curve database might have missing values or follows different sampling strategies for the sample size N. However, this remains an open question, as our learning curve database and experimental design do not explicitly address such case.

Table 3.6: Average ranking for extrapolation at $N = 100$ (lower is better) for both classification and regression problems.

	Classification			**Regression**		
	$Q = 10$	$Q = 25$	$Q = 50$	$Q = 10$	$Q = 25$	$Q = 50$
SPKR	**1.40**	**1.45**	**1.45**	**1.18**	**1.36**	**1.49**
MDS	1.60	1.55	1.55	1.82	1.64	1.51

Table 3.7: Average ranking for the whole curves (lower is better) for both classification and regression problems.

	Classification			**Regression**		
	$Q = 10$	$Q = 25$	$Q = 50$	$Q = 10$	$Q = 25$	$Q = 50$
SPKR	1.54	**1.40**	**1.28**	**1.29**	**1.44**	**1.38**
MDS	**1.46**	1.60	1.72	1.71	1.56	1.62

3.6 Conclusions

We introduced a data-driven approach that facilitates the rapid extrapolation of learning curves by incorporating already available learning curves. We utilize a learning curve database to extrapolate partially observed learning curves that are not present in the database. Our proposed method, called *SPKR*, extracts the relevant part of the dataset, applies dimensionality reduction and uses this information in combination with the partial observations to model the learning curves. To test our method, we create a learning curve database consisting of curves that have monotone and non-monotone behaviours. The extrapolation results demonstrate that, on average and rank wise, our approach yields better extrapolation performance than current parametric and non-parametric methods for learning curve extrapolation.

Although we show that SPKR outperforms the alternative approaches considered in the paper, it just provides a point estimate. In order to get an idea about the uncertainty of the estimate, its probabilistic counterpart, Gaussian Processes can be used (similar to [6]). This would be particularly useful for applications concerning learning curves such as hyperparameter optimization and model selection. Finally, we present our results using a densely sampled learning curve database without missing values. The effectiveness of our method in cases involving partially missing learning curves remains an open question. As a next step, investigating learning curves with varying sizes or partial observations presents an interesting research direction.

References

[1] Felix Mohr et al. "LCDB 1.0: An Extensive Learning Curves Database for Classification Tasks". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Massih-Reza Amini et al. Vol. 13717. Cham: Springer Nature Switzerland, 2023, pp. 3–19. DOI: 10.1007/978-3-031-26419-1_1.

[2] Tom Viering and Marco Loog. *The Shape of Learning Curves: A Review*. Nov. 2022. arXiv: 2103.10948 [cs].

[3] Marco Loog and Tom Viering. *A Survey of Learning Curves with Bad Behavior: or How More Data Need Not Lead to Better Performance*. 2022. arXiv: 2211.14061 [cs.LG].

[4] Steven Adriaensen et al. *Efficient Bayesian Learning Curve Extrapolation using Prior-Data Fitted Networks*. 2023. arXiv: 2310.20447 [cs.LG].

[5] Ethan Caballero et al. "Broken Neural Scaling Laws". In: *The Eleventh International Conference on Learning Representations*. 2023.

[6] Ethan Harvey et al. *A Probabilistic Method to Predict Classifier Accuracy on Larger Datasets given Small Pilot Data*. 2023. arXiv: 2311.18025 [cs.LG].

[7] Rui Leite and Pavel Brazdil. "Predicting relative performance of classifiers from samples". en. In: *Proceedings of the 22nd international conference on Machine learning - ICML '05*. Bonn, Germany: ACM Press, 2005, pp. 497–503. DOI: 10.1145/1102351.1102414.

[8] Zhiyi Chen, Marco Loog, and Jesse H. Krijthe. "Explaining Two Strange Learning Curves". In: *Artificial Intelligence and Machine Learning*. Ed. by Toon Calders et al. Cham: Springer Nature Switzerland, 2023, pp. 16–30.

[9] Marco Loog and Robert P. W. Duin. "The Dipping Phenomenon". In: *Structural, Syntactic, and Statistical Pattern Recognition*. Ed. by Georgy Gimel'farb et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 310–317.

[10] Tim Ruhkopf et al. "MASIF: Meta-learned Algorithm Selection using Implicit Fidelity Information". In: *Trans. Mach. Learn. Res.* 2023 (2023).

[11] Shayan Jawed et al. "Multi-task Learning Curve Forecasting Across Hyperparameter Configurations and Datasets". In: *Machine Learning and Knowledge Discovery in Databases. Research Track*. Ed. by Nuria Oliver et al. Cham: Springer International Publishing, 2021, pp. 485–501.

[12] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. "Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves". In: *International Joint Conference on Artificial Intelligence*. 2015.

[13] Aaron Klein et al. "Learning Curve Prediction with Bayesian Neural Networks". In: *International Conference on Learning Representations*. 2017.

[14] Romain Egele et al. *Is One Epoch All You Need For Multi-Fidelity Hyperparameter Optimization?* 2023. arXiv: 2307.15422 [cs.LG].

[15] Shen Yan et al. *NAS-Bench-x11 and the Power of Learning Curves*. en. arXiv:2111.03602 [cs]. Nov. 2021. DOI: 10.48550/arXiv.2111.03602.

[16] Dong Bok Lee et al. *Cost-Sensitive Multi-Fidelity Bayesian Optimization with Transfer of Learning Curve Extrapolation*. en. arXiv:2405.17918 [cs]. May 2024. DOI: 10.48550/arXiv.2405.17918.

[17] Claudia Perlich. "Learning Curves in Machine Learning". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 577–580. DOI: 10.1007/978-0-387-30164-8_452.

[18] Achin Jain et al. "A Meta-Learning Approach to Predicting Performance and Data Requirements". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 3623–3632. DOI: 10.1109/CVPR52729.2023.00353.

[19] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[20] Karl Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720.

[21] D. Billheimer. "Functional Data Analysis, 2nd Edition Edited by J. O. Ramsay and B. W. Silverman". In: *Biometrics* 63.1 (Apr. 2007), pp. 300–301. DOI: 10.1111/j.1541-0420.2007.00743_1.x. eprint: https://academic.oup.com/biometrics/article-pdf/63/1/300/52300836/biometrics_63_1_300.pdf.

[22] Peter Sollich. *Gaussian Process Regression with Mismatched Models*. 2001. arXiv: cond-mat/0106475 [cond-mat.dis-nn].

[23] Baohua Gu, Feifang Hu, and Huan Liu. "Modelling Classification Performance for Large Data Sets". In: *Advances in Web-Age Information Management*. Ed. by X. Sean Wang, Ge Yu, and Hongjun Lu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 317–328.

[24] Prasanth Kolachina et al. "Prediction of Learning Curves in Machine Translation". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Haizhou Li et al. Jeju Island, Korea: Association for Computational Linguistics, July 2012, pp. 22–30.

[25] Dong C. Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical Programming* 45 (1989), pp. 503–528.

[26] Lionel Kielhöfer, Felix Mohr, and Jan N. van Rijn. "Learning Curve Extrapolation Methods Across Extrapolation Settings". In: *Advances in Intelligent Data Analysis XXII*. Ed. by Ioanna Miliou, Nico Piatkowski, and Panagiotis Papapetrou. Cham: Springer Nature Switzerland, 2024, pp. 145–157.

[27] E. A. Nadaraya. "On Estimating Regression". In: *Theory of Probability & Its Applications* 9.1 (1964), pp. 141–142. DOI: 10.1137/1109020.

[28] Harald Steck, Chaitanya Ekanadham, and Nathan Kallus. "Is Cosine-Similarity of Embeddings Really About Similarity?" In: *Companion Proceedings of the ACM Web Conference 2024*. WWW '24. Singapore, Singapore: Association for Computing Machinery, 2024, pp. 887–890. DOI: 10.1145/3589335.3651526.

[29] Karl Bringmann et al. *Dynamic Dynamic Time Warping*. 2023. arXiv: 2310.18128 [cs.CG].

[30] Rui Leite and Pavel Brazdil. "Active Testing Strategy to Predict the Best Classification Algorithm via Sampling and Metalearning". In: *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*. NLD: IOS Press, 2010, pp. 309–314.

3

4 | On Sample-Wise Strict Monotonicity with a Gradient Update

Learning curves describe how the performance of a model evolves with increasing training data. Although more data is generally expected to improve model performance, in practice models can exhibit non-monotonic behavior where additional data leads to performance degradation. Sample-wise double descent is one particular example. We address the question of how a learner can have a provably monotone learning curve. For isotropic Gaussian covariates under a Gaussian noise model and a linear predictor, we prove that a single step of steepest descent guarantees sample-wise monotonicity in the learning curve, if the step size does not exceed an upper bound. Furthermore, we present a practical procedure that ensures monotonicity without explicit regularization or cross-validation, using initialization from the previous training set size. Experiments on real-world datasets demonstrate that this method achieves monotone behavior and improved sample efficiency compared to ordinary least squares and optimally regularized ridge regression. We also explore extensions to binary classification, where monotonicity depends on the chosen performance metric. While our guarantees are derived under simplifying assumptions, they provide both theoretical and practical insights for constructing monotone learners and for understanding and mitigating sample-wise double descent behavior.

4.1 Introduction

Does additional data always help? Intuitively, it should, but in practice, for some problems and some models, it does not. There are classification and regression problems both in supervised [1] and in semi-supervised learning [2] where the expected performance can get worse with additional training data. In general, it is desirable to have models get better with data, as it makes it easier to predict, for a given level of model performance, how much more data should be collected. This dependence of expected performance on training set size is studied using learning curves. Learning curves, which plot expected performance as a function of training set size, should not be confused with training curves commonly used in the deep learning literature,

where performance is shown as a function of training epochs.

Although this observation of non-monotonic behaviour dates back to 1989 [1], its connection to deep learning [3] and to questions of model complexity renewed interest in it. In [4], the double descent phenomenon with respect to model complexity is examined in detail for linear regression, trees, and boosting. Their analysis shows that the conclusions drawn in [3] were rather flawed, because multiple axes of complexity were conflated in the original study.

We are concerned with the sample-wise double descent phenomenon for linear regression which exhibits an unconventional bias-variance trade-off: as the number of samples increases, the bias decreases while the variance increases [5]. Despite this atypical behavior, optimal ridge regularization has been shown to mitigate the effect [6].

4

We show that for ordinary least squares (OLS) the problem of double descent along learning curves can also be mitigated without explicitly regularizing the loss function. First, we prove that for an initial parameter vector and step size independent of the training points, if only one gradient step is taken, the expected performance decreases monotonically provided that the step size does not exceed a certain value. We also present, for Gaussian covariates, what this maximum step size is that ensures sample-wise monotonicity. Although this proof is under strict assumptions about the data-generating process, we propose a way to make monotone linear regression models. In this approach, we still take one gradient update, but with exact line search starting from the parameters of the average estimator obtained from the previous training set size.

An illustration of our method is shown in Figure 4.1, which displays the learning curves of several models on isotropic Gaussian data. In this plot, the green curve corresponds to our proposal, where only one gradient update is performed using exact line search, starting from the average estimator of the previous training set size (a warm start). The orange curve depicts the optimally regularized ridge estimator proposed in [6], and the blue curve shows the standard OLS estimator. As seen in the same figure, the OLS estimator exhibits sample-wise double descent behavior, which disappears under optimal regularization. Likewise, our proposed estimator does not display this behavior.

Similar efforts on making general learners more monotone have been undertaken in [7] and [8]. Both of these works make the learners monotone via a smart selection strategy. Our proposal is not model-agnostic as we restrict the model to linear regression. We are only focused on making linear regression problems monotone.

Published as: **O. Taylan Turan** et al. "On Sample-Wise Strict Monotonicity with a Gradient Update". In: Advances in Intelligent Data Analysis XXIV (2026).

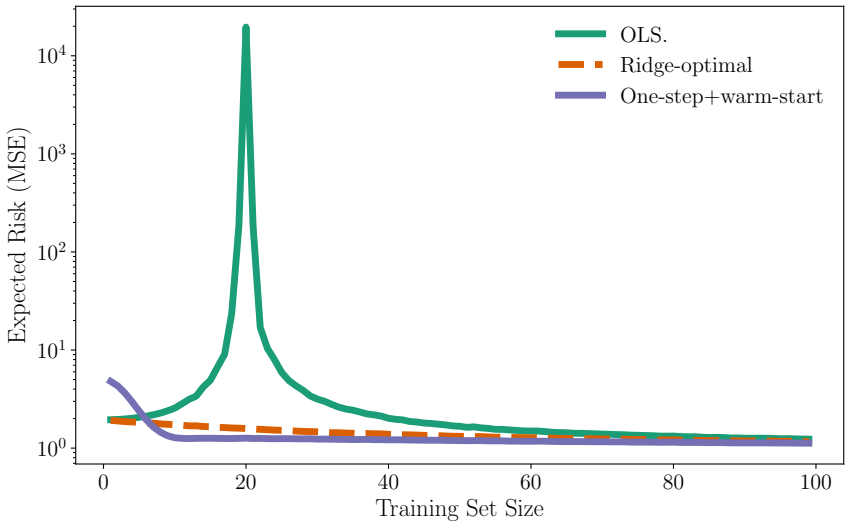


Figure 4.1: Expected MSE with respect to the training set size n, where inputs come from an isotropic Gaussian $\boldsymbol{x} \sim \mathcal{N}(0, I_d)$ and outputs follow $y = {\boldsymbol{\beta}^*}^\top \boldsymbol{x} + \mathcal{N}(0, \sigma^2)$, with $d = 20$, $\sigma^2 = 1$, and $\|\boldsymbol{\beta}^*\|_2 = 1$.

Moreover, we do not require a strategy other than using the previous training set size for the initial guess. Hence, the closest work to our effort is [6], where, similarly, a supervised linear regression problem is proven to be monotone with optimal regularization. In practice, the optimal regularization parameter is of course not available and one needs to resorts to techniques like cross-validation to estimate it.

Our paper is organized as follows. Section 4.2 introduces expected risk and monotonicity along learning curves for the linear regression problem, and we show that for the optimal step size, the expected risk decreases monotonically. In Section 4.3, we make use of our theoretical findings to create monotone learners in practice. Finally, we reflect on our findings and point out open questions.

4.2 Strict Monotonicity for One-Step Gradient Update

We consider the linear regression problem where the input $\boldsymbol{x} \in \mathbb{R}^d$ is from a random distribution and the output (or response) is generated according to the model:

$$y = {\boldsymbol{\beta}^*}^\top \boldsymbol{x} + \varepsilon, \tag{4.1}$$

where ε is random noise, and $\boldsymbol{\beta}^* \in \mathbb{R}^d$ is an unknown parameter vector. Let $\mathcal{D}$ denote the joint distribution of $(\boldsymbol{x}, y)$. In practice, we are given a training dataset

$\mathcal{D}_n := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, drawn independently from $\mathcal{D}$. The goal is to learn a linear model $f_{\boldsymbol{\beta}}(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{x}$, with $\boldsymbol{\beta} \in \mathbb{R}^d$ that minimizes the population mean-squared error:

$$\mathcal{R}(\boldsymbol{\beta}) := \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{D}} \left[\left(\boldsymbol{\beta}^\top \boldsymbol{x} - y \right)^2 \right]. \tag{4.2}$$

Let's assume $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ to be the data matrix whose rows are the covariates $\mathbf{x}_i^\top$, and let $\boldsymbol{y} \in \mathbb{R}^n$ be the corresponding vector of responses. For any estimator $\hat{\boldsymbol{\beta}}_n = \boldsymbol{\beta}_n(\boldsymbol{X}, \boldsymbol{y})$ expected risk is defined as:

$$\bar{\mathcal{R}}_n := \mathop{\mathbb{E}}_{\boldsymbol{X},\boldsymbol{y}\sim\mathcal{D}} \left[\mathcal{R}(\hat{\boldsymbol{\beta}}_n) \right]. \tag{4.3}$$

Assume input data comes from a zero mean isotropic Gaussian distribution $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$ and the noise is homoscedastic and Gaussian $\varepsilon \sim \mathcal{N}\left(0, \sigma^2\right)$. We consider ordinary least-squares estimator (OLS) with one gradient update step. Recall that the OLS estimator is given by

$$\hat{\boldsymbol{\beta}}_n := \arg\min_{\boldsymbol{\beta}} \left\{ \|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 \right\}. \tag{4.4}$$

Defining the Gram matrix as $\boldsymbol{\Sigma}_n = \boldsymbol{X}^\top \boldsymbol{X}$, the scaled gradient for the OLS objective is given by,

$$\boldsymbol{r}_n := \boldsymbol{X}^\top \boldsymbol{y} - \boldsymbol{\Sigma}_n \bar{\boldsymbol{\beta}}. \tag{4.5}$$

If $\bar{\boldsymbol{\beta}}$ is the initial guess, then our estimator takes the following form:

$$\hat{\boldsymbol{\beta}}_n = \bar{\boldsymbol{\beta}} + \alpha \boldsymbol{r}_n, \tag{4.6}$$

where α is the step size. Finally, we assume that initial guess $\bar{\boldsymbol{\beta}}$ and step size α are independent of covariates and corresponding responses. Our starting point is to obtain the expected risk given in Equation 4.3 with $\hat{\boldsymbol{\beta}} = \bar{\boldsymbol{\beta}} + \alpha(\boldsymbol{X}^\top \boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{X} \bar{\boldsymbol{\beta}})$ and we represent the difference between the initial guess and the optimal parameters as $\widetilde{\boldsymbol{\beta}} = \bar{\boldsymbol{\beta}} - \boldsymbol{\beta}^*$.

Theorem 1. *For the above setting, the expected risk of well-specified isotropic linear regression problem is sample-wise monotone, that is $\bar{\mathcal{R}}_{n+1} \leq \bar{\mathcal{R}}_n$, if $0 \leq \alpha \leq \alpha_{max}$, where*

$$\alpha_{max} := \frac{2\|\widetilde{\boldsymbol{\beta}}\|_2^2}{d\sigma^2 + (2 + d + 2n)\|\widetilde{\boldsymbol{\beta}}\|_2^2}.$$

Lemma 1. *For the above setting, the expected risk can be represented as a function of $\alpha, \bar{\boldsymbol{\beta}}, \boldsymbol{\beta}^*, d$*

and n as,

$$\bar{\mathcal{R}}_n = (\alpha^2(n^2+n+dn) - 2\alpha n + 1)\|\widetilde{\boldsymbol{\beta}}\|_2^2 + (\alpha^2 nd + 1)\sigma^2$$

Proof of Lemma 1. For isotropic Gaussian covariates the risk of an estimator for a given n training samples is $\mathcal{R}_n := \|\hat{\boldsymbol{\beta}}_n - \boldsymbol{\beta}^*\|_2^2 + \sigma^2$. Then, the expected risk for any given training set size n can be obtained as follows [6]:

$$\mathcal{R}_n = \|\hat{\boldsymbol{\beta}}_n - \boldsymbol{\beta}^*\|_2^2 + \sigma^2$$

Using our one-step steepest descent estimator $\boldsymbol{\beta}^*$ from the initial guess $\bar{\boldsymbol{\beta}}$,

$$\begin{aligned}
\bar{\mathcal{R}}_n &= \mathop{\mathbb{E}}_{\boldsymbol{X},\boldsymbol{y}}\left[\|\alpha\boldsymbol{X}^\top\boldsymbol{y} - \alpha\boldsymbol{X}^\top\boldsymbol{X}\bar{\boldsymbol{\beta}} + \widetilde{\boldsymbol{\beta}}\|_2^2\right] + \sigma^2 \\
&= \mathop{\mathbb{E}}_{\boldsymbol{X},\boldsymbol{\eta}\sim\mathcal{N}(\boldsymbol{0},\sigma^2\boldsymbol{I}_n)}\left[\|\alpha\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\beta}^* + \boldsymbol{\eta}) - \alpha\boldsymbol{X}^\top\boldsymbol{X}\bar{\boldsymbol{\beta}} + \widetilde{\boldsymbol{\beta}}\|_2^2\right] + \sigma^2 \\
&= \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{X}}\left[\|\alpha\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta}^* - \alpha\boldsymbol{X}^\top\boldsymbol{X}\bar{\boldsymbol{\beta}} + \widetilde{\boldsymbol{\beta}}\|_2^2\right]}_{\text{A}} + \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{X},\boldsymbol{\eta}}\left[\|\alpha\boldsymbol{X}^\top\boldsymbol{\eta}\|_2^2\right]}_{\text{B}} + \sigma^2
\end{aligned}$$

Since the elements of $\boldsymbol{X}$ are drawn from an isotropic Gaussian distribution ($\mathcal{N}(0, \mathbf{I}_d)$), the Gram matrix is a Wishart distribution $\boldsymbol{\Sigma} = \boldsymbol{X}^\top\boldsymbol{X} \sim \mathcal{W}(\mathbf{I}_d, d)$. The first and the second moments of Wishart distribution are given by

$$\begin{aligned}
&\mathop{\mathbb{E}}_{X}[\boldsymbol{\Sigma}] = n\mathbf{I}_d, \quad \text{and} \\
&\mathop{\mathbb{E}}_{X}[\boldsymbol{\Sigma}\boldsymbol{\Sigma}] = \mathop{\mathbb{E}}_{X}\left[\boldsymbol{\Sigma}^2\right] = nd\mathbf{I}_d + (n^2+n)\mathbf{I}_d,
\end{aligned}$$

respectively [9]. Expanding the squared norm in A and using the expressions for moments of the Wishart distribution leads to

$$\begin{aligned}
\text{A} &= \alpha\mathop{\mathbb{E}}_{\boldsymbol{X}}\left[\alpha\boldsymbol{\beta}^{*\top}\boldsymbol{\Sigma}^2\boldsymbol{\beta}^* - 2\alpha\boldsymbol{\beta}^{*\top}\boldsymbol{\Sigma}^2\bar{\boldsymbol{\beta}} + 2\boldsymbol{\beta}^{*\top}\boldsymbol{\Sigma}\widetilde{\boldsymbol{\beta}} + \alpha\bar{\boldsymbol{\beta}}^\top\boldsymbol{\Sigma}^2\bar{\boldsymbol{\beta}} - 2\bar{\boldsymbol{\beta}}^\top\boldsymbol{\Sigma}\widetilde{\boldsymbol{\beta}}\right] + \widetilde{\boldsymbol{\beta}}^\top\widetilde{\boldsymbol{\beta}} \\
&= \alpha^2(n^2+n+dn)[\|\boldsymbol{\beta}^*\|_2^2 - 2\boldsymbol{\beta}^{*\top}\bar{\boldsymbol{\beta}} + \|\bar{\boldsymbol{\beta}}\|_2^2] + 2\alpha n(\boldsymbol{\beta}^{*\top}\widetilde{\boldsymbol{\beta}} - \bar{\boldsymbol{\beta}}^\top\widetilde{\boldsymbol{\beta}}) + \|\widetilde{\boldsymbol{\beta}}\|_2^2 \\
&= (\alpha^2(n^2+n+dn) - 2\alpha n + 1)\|\widetilde{\boldsymbol{\beta}}\|_2^2.
\end{aligned}$$

We can obtain the B as

$$\begin{aligned}
\text{B} &= \mathop{\mathbb{E}}_{\boldsymbol{X},\boldsymbol{\eta}}\left[\|\alpha\boldsymbol{X}^\top\boldsymbol{\eta}\|_2^2\right] = \mathop{\mathbb{E}}_{\boldsymbol{X}}\left[\mathop{\mathbb{E}}_{\boldsymbol{\eta}}\left[\|\alpha\boldsymbol{X}^\top\boldsymbol{\eta}\|_2^2|\boldsymbol{X}\right]\right] \\
&= \alpha^2\sigma^2\mathop{\mathbb{E}}_{\boldsymbol{X}}\left[\|\boldsymbol{X}\|_{\text{F}}^2\right] = \alpha^2\sigma^2\text{tr}(\mathop{\mathbb{E}}_{\boldsymbol{X}}[\boldsymbol{\Sigma}]) = \alpha^2\sigma^2 nd.
\end{aligned}$$

Then, the expected performance at training set size n can be expressed as,

$$\bar{\mathcal{R}}_n = (\alpha^2(n^2+n+dn) - 2\alpha n + 1)\|\widetilde{\boldsymbol{\beta}}\|_2^2 + \alpha^2\sigma^2 nd + \sigma^2$$

Now, we provide the proof for Theorem 1.

Proof of Theorem 1. Using Lemma 1, $\bar{\mathcal{R}}_{n+1} \leq \bar{\mathcal{R}}_n$ takes the form after some rearranging,

$$\bar{\mathcal{R}}_{n+1} - \bar{\mathcal{R}}_n := \underbrace{[(2n+2+d)\|\widetilde{\boldsymbol{\beta}}\|_2^2 + \sigma^2 d]}_{a}\,\alpha^2 - \underbrace{2\|\widetilde{\boldsymbol{\beta}}\|_2^2}_{b}\,\alpha \leq 0, \tag{4.7}$$

which is quadratic in the step size $a\alpha^2 - b\alpha \leq 0$ with respect to step size. Since we want to update the initial guess with one gradient update, step size must be bigger than 0 $\alpha > 0$. Then, the solution can be found as;

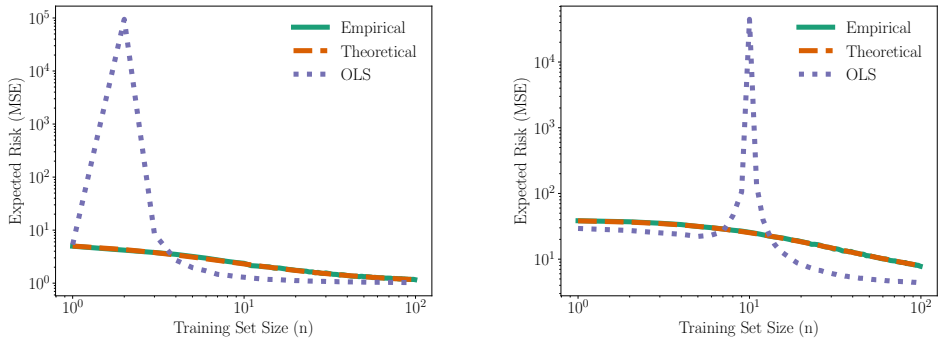
$$\left(b < 0 \;\wedge\; a < 0 \;\wedge\; \alpha \geq \frac{b}{a}\right) \vee$$
$$\left(b \geq 0 \;\wedge\; \left((a \leq 0 \;\wedge\; \alpha > 0) \;\vee\; (a > 0 \;\wedge\; 0 < \alpha \leq \frac{b}{a})\right)\right).$$

Since, $b < 0$ and $a < 0$ is not possible values for our case given in Equation 4.7, the only valid interval for our case can be found as $0 < \alpha \leq \frac{b}{a}$.

To support our theoretical findings, in Figure 4.2 we show for various dimensionalities and noise-levels, the sample-wise monotonicity with respect to OLS estimator and the alignment of our theoretical results with empirical observations. These plots are obtained with the exact setting described above. In the figure the green line represents the empirical expected risk for mean squared error metric and the dotted orange line shows Theorem 1. For the empirical results, we repeat the sampling of the training set 1000 times. As can be observed, our empirical findings using the maximum step size are closely aligned with our theoretical findings, when one gradient update is used for training with maximum step size α_{max}. These observations are consistent among different dimensionality and noise-variance.

4.3 Monotonicity in Practice

Obtaining the expected risk given in Equation 4.3 in a real-world setting is impossible since $\mathcal{D}$ is not known. Generally, bootstrapping or cross-validation methods are employed to get an estimate of the expected risk while creating learning curves [10]. To calculate the expected risk empirically, we first randomly sample 20% of the dataset for the test set. Then, the remaining 80% of the dataset containing N_{train} samples is used for training. In our experiments, we sample a new training set without replacement for each training set size 1000 times, in order to get a reliable estimate of



(a) $d = 2, \quad \sigma^2 = 1$ (b) $d = 10, \quad \sigma^2 = 4$

Figure 4.2: Expected MSE with respect to training set size. We simulate the setting of Theorem 1 and use the expression derived in Lemma 1 with the maximum step size α_{max}. In this simulation, the data-generating parameters are drawn from a normal distribution $\boldsymbol{\beta}^* \sim \mathcal{N}(0, 1)$, and the initial parameter vector is also drawn independently from a normal distribution $\bar{\boldsymbol{\beta}} \sim \mathcal{N}(0, 1)$.

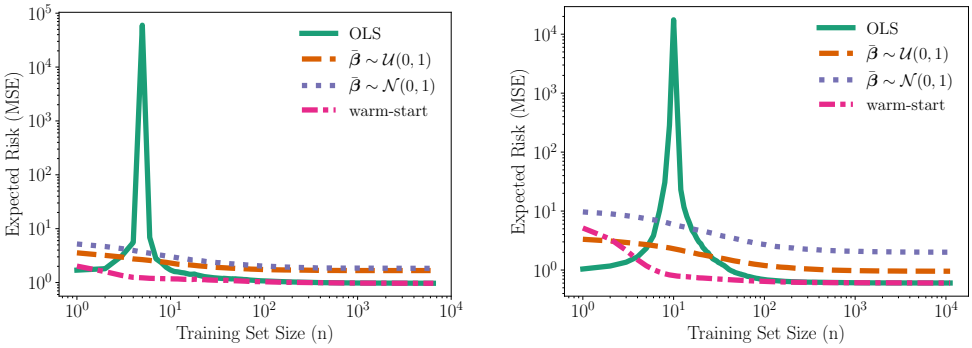
the expected risk. We repeat the same procedure for every sample size in the range $n \in [1, 0.8 \times N_{\text{train}}]$ to create the learning curves. Finally, for training, we normalize both the features and the outputs.

Note that the maximum step size α_{max} found in Theorem 1 depends on the variance of the noise σ^2 and the norm of the difference between the initial guess for the estimator and the optimal parameter vector $\|\widetilde{\boldsymbol{\beta}}\|_2$, which means it depends on the data generating process. To make our proposal applicable in practice, we propose using exact line search since our objective function is convex. Then, the step size for our estimator with n training samples takes the form $\alpha_{\text{ls}} := \frac{\boldsymbol{r}_n^\top \boldsymbol{r}_n}{\boldsymbol{r}_n^\top \boldsymbol{\Sigma}_n \boldsymbol{r}_n}$.

In Figure 4.3, we present results for two datasets obtained from OpenML [11]. For each dataset, we consider two models with fixed initializations: one drawn from a normal distribution and the other from a uniform distribution. Each model performs a single gradient update using exact line search, which results in monotonic learning curves. We observe that the expected performance depends on the quality of the fixed initialization. Moreover, convergence to the ordinary least squares (OLS) solution may not occur as the training set size increases.

To mitigate the effect of the starting point, we propose a warm start for each training set size n by using the average solution obtained using $n - 1$ samples on top of the exact line search with one gradient update. We cannot have a warm start for the first sample, and we initialize $\bar{\boldsymbol{\beta}}_1$ with a random sample from a standard normal dis-

Code used for the experiments is available at https://github.com/taylanot/monotonicity.git



(a) Sulfur dataset

(b) Bike sharing dataset

Figure 4.3: Learning curves for two OpenML [11] datasets, Sulfur and Bike Sharing. We compare fixed initial guesses for the parameter vector (sampled from uniform and normal distributions) with a warm start, all using a single gradient update with exact line search.

tribution. For the rest of the training set sizes, the average parameter vector from the previous training set size will be used; $\bar{\boldsymbol{\beta}}_n := \frac{1}{m}\sum_{i=1}^{m}\hat{\boldsymbol{\beta}}_{n-1}$, where m is the number of training sets drawn from a given training set size for obtaining $\bar{\mathcal{R}}_n$. Then, the proposed estimator for training set size n takes the form,

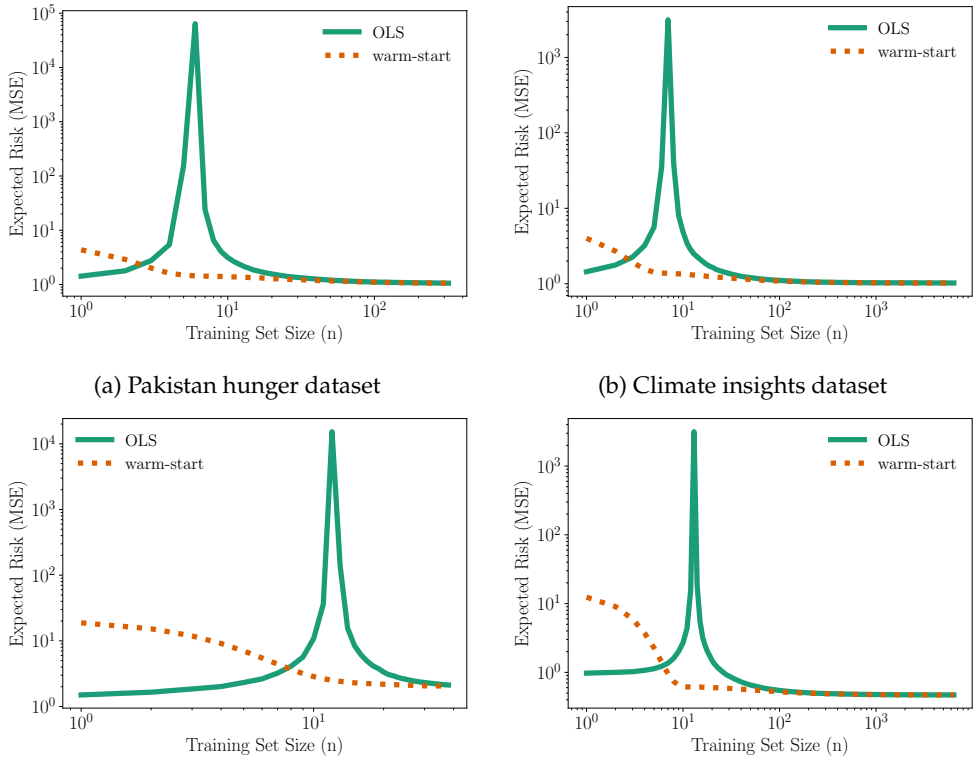
$$\hat{\boldsymbol{\beta}}_n = \bar{\boldsymbol{\beta}}_n + \alpha_{\text{ls}}\boldsymbol{r}_n \tag{4.8}$$

As can be seen from Figure 4.3, the warm starting point makes the solution converge to the OLS solutions as the training set size increases. We observed the same in four more random supervised regression problems from OpenML [11] in Figure 4.4, where the green line represents the OLS solution that gives the minimum-norm solution when $d \geq n$ and the orange dotted line represents our proposed solution using one gradient update, exact line search and warm initialization.

For all the datasets considered, our estimator is monotone, where the sample-wise double descent is not present. Moreover, our proposal converges to the OLS solution that is achieved after the double descent rapidly in terms of sample-size needed. Finally, we observe that this monotonicity does not always come at the cost of low performance as well.

4.4 Discussion and Conclusion

We tried to answer one of the open questions presented in [12]: when is a learner provably monotone? We proved that for isotropic Gaussian covariates, a linear regression with a Gaussian noise model and one step of steepest descent can result in a sample-



(a) Pakistan hunger dataset

(b) Climate insights dataset

(c) Sleep deprivation dataset

(d) Climate change impact dataset

Figure 4.4: Learning curves for four other OpenML [11] datasets: Pakistan hunger, climate insights, sleep deprivation, and climate change impact. Across all learning curves, our proposed approach exhibits monotonic behaviour compared to its OLS counterparts.

wise monotone learning algorithm, given that the maximum step size is not exceeded. We also provide this limiting step size for which this monotonicity holds. However, in practice, we of course typically do not know the optimal step size. Hence, we propose a practical way to set this size. We showed on real-world datasets that if one step of steepest descent is used with fixed parameters, it yields sample-wise monotonic learner.

Our proof relies on three major assumptions. The initial parameter vector and step size are independent of the training points and, the training points are drawn from an isotropic Gaussian distribution with Gaussian noise model. Both the warm start and the exact line search we suggest to be used in practice violate the independence assumption. We showed empirically that both the step size dependence alone and the combined warm starting and step size dependence does not violate the monotonicity

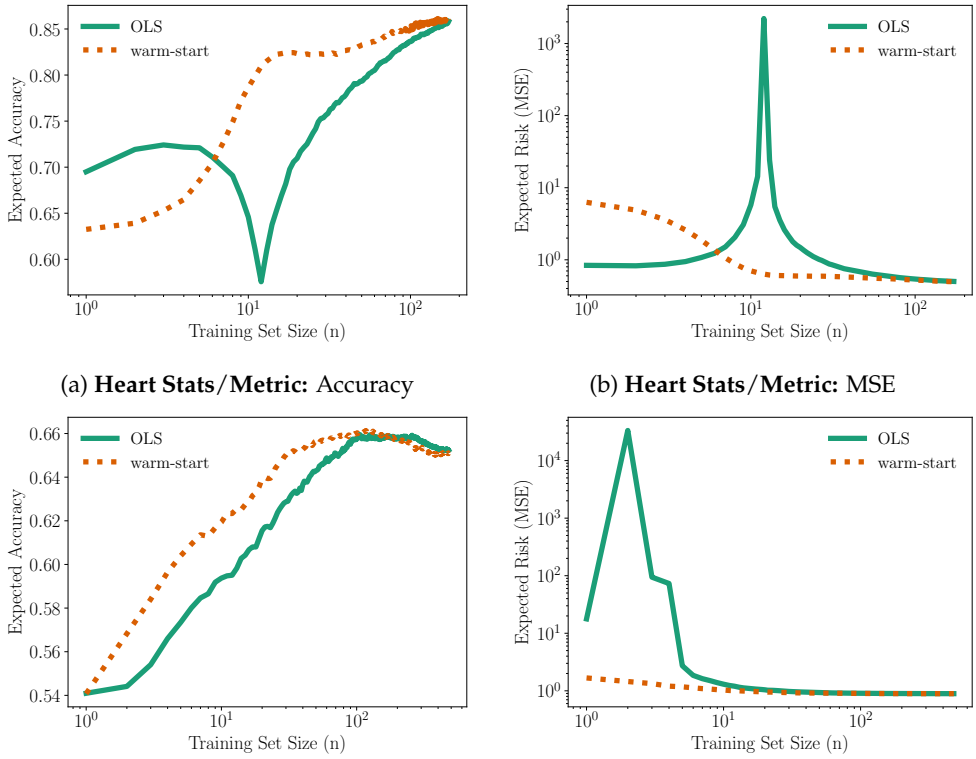
in practice. A theoretical analysis may be difficult when both parameters are allowed to depend on the covariates. We suspect that, although the exact line search decision for the step size introduces dependence on the training points, it may still be possible to bound the expected step size to guarantee sample-wise monotonicity. However, this may not be possible for the warm start initialization. It may be easier to first investigate whether relaxing the Gaussian assumption on the covariates enables a similar analysis.

Unlike other regularization techniques resulting in monotonicity, our practical proposal of one step of steepest descent does not require any cross-validation. Instead, it relies on a single gradient update with an analytically derived step size. This eliminates the need for hyperparameter search or data splitting, both of which can introduce additional sources of variance while estimating the expected risk. The absence of tuning makes our approach simple to implement, and well-suited for scenarios with limited data, where validation sets are limited. Despite its simplicity, the resulting estimator exhibits performance comparable to optimally regularized models. However, to have performance converging to OLS estimate in practice our approach relies on the average parameters from the previous training set sizes $n-1$, which introduces a computational complexity that is essentially equal to that of leave one out. A sparser grid for the training set size can reduce this burden, although it may also result in average solution from the previous training set sizes to be further apart from the current one, which can lead to non-monotonic behaviour.

We investigated only the monotonicity aspect. However, as shown in Figures 4.1 and 4.4, our method is not only monotone but also stabilizes more quickly to a lower generalization error as the training set size increases, compared to the other methods. Similar observations, where *gradient descent* outperforms the analytical OLS solution, are also reported in [13].

All of our results pertain to the regression setting with the MSE loss. When other measures are used for obtaining the learning curve, some non-monotone behaviour can be observed (although not as severe as for the OLS). In Figure 4.5 we allude to such cases for two random binary classification problems (multi-class classification problems, require additional decisions to determine the warm starting point of the parameter vector), with same experimental setting apart from the loss measure provided in Section 4.3. The non-monotonic behavior of our approach could stem from the mismatch between the MSE considered during training, while testing is done using accuracy. [14, 10] have more generally indicated that metric mismatch between training and testing can introduce unexpected behavior. Expected error of Pseudo Fisher Linear Discriminant (PFLD) is presented in [15], where also analytically, the peaking behaviour [16] is observed. Our proposal addresses the non-monotonic be-

haviour introduced by the pseudo-inverse operation in the OLS estimator. For this reason, it may also be possible to guarantee monotonicity for linear classifiers that rely on the same pseudo-inverse, such as PFLD, for error rate or accuracy.



(a) **Heart Stats/Metric:** Accuracy

(b) **Heart Stats/Metric:** MSE

(c) **Blood Trans./Metric:** Accuracy

(d) **Blood Trans./Metric:** MSE

Figure 4.5: The effect of metric choice (Accuracy vs. MSE) across two datasets (heart stats and blood transfusion) from OpenML [11]. Each row corresponds to a dataset, with the left column showing expected Accuracy and the right column showing expected MSE as functions of training set size.

All in all, we have given a case when a learner is provably monotone in a constrained setting. However, our observations provide no universal guarantees: there might exist a distribution $\mathcal{D}$ for which any estimator performs poorly even for arbitrarily large training sizes n [17]. Thus, outside the narrow regimes where we can guarantee monotonicity, the very existence of a universally monotone learners remains unknown. For the MSE, one may even wonder if such learner at all exists.

References

[1] Marco Loog et al. "A brief prehistory of double descent". In: *Proceedings of the National Academy of Sciences* 117.20 (2020), pp. 10625–10626. DOI: 10.1073/pnas.2001875117. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2001875117.

[2] Jesse H. Krijthe and Marco Loog. "The Peaking Phenomenon in Semi-Supervised Learning". In: *Structural, Syntactic, and Statistical Pattern Recognition*. Ed. by Antonio Robles-Kelly et al. Cham: Springer International Publishing, 2016, pp. 299–309.

[3] Mikhail Belkin et al. "Reconciling Modern Machine-Learning Practice and the Classical Bias–Variance Trade-Off". In: *Proceedings of the National Academy of Sciences* 116.32 (Aug. 2019), pp. 15849–15854. DOI: 10.1073/pnas.1903070116.

[4] Alicia Curth, Alan Jeffares, and Mihaela van der Schaar. *A U-turn on Double Descent: Rethinking Parameter Counting in Statistical Learning*. Oct. 2023. arXiv: 2310.18988 [cs, stat].

[5] Preetum Nakkiran. *More Data Can Hurt for Linear Regression: Sample-wise Double Descent*. Dec. 2019. arXiv: 1912.07242 [cs, math, stat].

[6] Preetum Nakkiran et al. *Optimal Regularization Can Mitigate Double Descent*. Apr. 2021. arXiv: 2003.01897 [cs, math, stat].

[7] Tom Julian Viering, Alexander Mey, and Marco Loog. "Making Learners (More) Monotone". In: *Advances in Intelligent Data Analysis XVIII*. Ed. by Michael R. Berthold, Ad Feelders, and Georg Krempl. Cham: Springer International Publishing, 2020, pp. 535–547.

[8] Olivier J Bousquet et al. "Monotone Learning". In: *Proceedings of Thirty Fifth Conference on Learning Theory*. Ed. by Po-Ling Loh and Maxim Raginsky. Vol. 178. Proceedings of Machine Learning Research. PMLR, Feb. 2022, pp. 842–866.

[9] Melinda Hagedorn. *Expected Value of Matrix Quadratic Forms with Wishart distributed Random Matrices*. 2022. arXiv: 2212.01412 [math.OC].

[10] Tom Viering and Marco Loog. *The Shape of Learning Curves: A Review*. Nov. 2022. arXiv: 2103.10948 [cs].

[11] Bernd Bischl et al. "OpenML: Insights from 10 years and more than a thousand papers". In: *Patterns* 6.7 (2025), p. 101317. DOI: 10.1016/j.patter.2025.101317.

[12] Tom Viering, Alexander Mey, and Marco Loog. "Open Problem: Monotonicity of Learning". In: *Proceedings of the Thirty-Second Conference on Learning Theory*. PMLR, June 2019, pp. 3198–3201.

[13] Jingfeng Wu et al. *Risk Comparisons in Linear Regression: Implicit Regularization Dominates Explicit Regularization*. Sept. 2025. DOI: 10.48550/arXiv.2509.17251. arXiv: 2509.17251 [stat].

[14] Marco Loog and Robert P. W. Duin. "The Dipping Phenomenon". In: *Structural, Syntactic, and Statistical Pattern Recognition*. Ed. by Georgy Gimel'farb et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 310–317.

[15] Sarunas Raudys and Robert P. W. Duin. "Expected Classification Error of the Fisher Linear Classifier with Pseudo-Inverse Covariance Matrix". In: *Pattern Recognition Letters* 19.5 (Apr. 1998), pp. 385–392. DOI: 10.1016/S0167-8655(98)00016-6.

[16] Robert PW Duin. "Small sample size generalization". In: *Proceedings of the Scandinavian Conference on Image Analysis*. Vol. 2. 1995, pp. 957–964.

[17] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. 3. print. Applications of Mathematics 31. New York, NY: Springer, 2008.

4

5 | When MAML Learns Quickly Does it Generalize Well?

Model-Agnostic Meta-Learning is a meta-learning method that achieved state-of-the-art performance on few-shot image classification benchmarks at the time of its introduction. MAML's strength is its ability to quickly adapt to a new task in time-critical settings, while still being general enough for any gradient-based model. Although there is no need for quick adaptation in most of the few-shot learning benchmarks, often MAML is utilized as a benchmark in this context. We investigate the benefit of limiting the adaptation steps of MAML in settings where quick adaptation is not required by the problem. In this initial study, the expected performance of MAML is compared to some conventional base learners for synthetic linear and nonlinear regression problems. Our experimental results show that limited gradient descent steps only improve generalization performance when faced with small task variance.

5.1 Introduction

Learning to learn, also referred to as meta-learning, treats the training of a machine learning model as a learning problem in itself. In this setting, there exist multiple learning problems and they are treated together. A machine learning model is "learning to learn" if the performance on each task improves with training experience obtained from the other tasks [1]. A major use of meta-learning is to tackle few-shot learning problems, where there is little data available from the learning task that is of prime interest, whereas there is an abundance of data from other, yet similar tasks.

Early works on the learning-to-learn paradigm relied upon two different models working together, where one model tries to improve performance on the specific task and the other tries to improve performance over the observed tasks together [1]. MAML (Model-Agnostic Meta-Learning) [2] provides an algorithm that circumvents the need for multiple models. This method tackles meta-learning by providing a model initialization (which is always required for models that are optimized with Stochastic Gradient Descent) that facilitates quick adaptation and good generalization.

Presented as: **O. Taylan Turan** et al. "When MAML Learns Quickly, Does It Generalize Well?" In: BNAIC BeNeLearn (2022).

Since it is model and problem independent, MAML finds a wide application area in the context of few-shot meta-learning. Specifically, for supervised and reinforcement learning problems under that paradigm. Moreover, MAML also aims to improve a specific task performance quickly (with a few gradient steps). This is an additional important aspect of meta-learning.

The quick adaptation feature can prove useful in certain settings, for instance, in robotics applications, where the reaction/adaptation time of the agents in dynamic environments imposes time limitations. However, this limitation is not present in supervised learning problems, where MAML or its variants are utilized as a baseline (*e.g.* [3, 4, 5, 6, 7]). Most supervised problems are benchmarked with an image classification problem, where N-way K-shot classification problem (N different classes with K labeled training data) is tackled, where memory or time limitations do not constitute a major issue.

The main aim of this paper is to investigate MAML in settings where quick adaptation is not needed, and where most of the applications and variants of this method are benchmarked. This will be achieved by looking at the expected performance of MAML under two synthetic regression scenarios, and comparing its performance to conventional base learners (*e.g.* Linear Regression, Ridge Regression, Kernel Ridge Regression, etc.). By doing this we aim to investigate the effect of the limited adaptation step, and whether or not there is a benefit to this limitation. The code for all experiments is available at github.com/taylanot/EE_MAML.git.

5.2 Model-Agnostic Meta-Learning (MAML)

MAML aims to obtain an intermediate model $\mathcal{M}(\bar{\mathbf{w}}_{\text{meta}})$ that can generalize well after adaptation with gradient descent to a dataset $\mathcal{Z}$ observed from a new and unseen task $\mathcal{T}$ drawn from $p_{\mathcal{T}}$ where the number of training points N and the number of iterations n_{iter} is limited.

In order to obtain $\bar{\mathbf{w}}_{\text{meta}}$ first, a batch of tasks $\{\mathcal{T}_i\}_{i=1}^{M}$ from $p_{\mathcal{T}}$ is observed with each having a corresponding dataset $\{\mathcal{Z}_i\}_{i=1}^{M}$. Then, the future gradients concerning each task are observed and gradient descent is utilized to get possible parameters $\bar{\mathbf{w}}'$ for each task and a gradient descent iteration is made by collecting all the possible parameters from an observed batch of tasks for the real parameter update. The general procedure for supervised learning problems is given in Algorithm 2. Authors of [2] indicate that by using this procedure the model $\mathcal{M}(\bar{\mathbf{w}}_{\text{meta}})$ requires few gradient updates from a specific task, achieving good generalization performance. Examples of the intermediate model $\mathcal{M}(\bar{\mathbf{w}}_{\text{meta}})$ prediction with the observed tasks on the background can be seen in Figures 5.1a and 5.1b.

Data: $p_{\mathcal{T}}$, α, β
Result: Intermediate Model $\mathcal{M}(\bar{\mathbf{w}}_{meta})$
initialize $\bar{\mathbf{w}}$ randomly;
while *not done* **do**
  sample a batch of tasks $\mathcal{T}_i$ from $p_{\mathcal{T}}$
  forall $\mathcal{T}_i$ **do**
    Obtain future gradients: $\nabla_{\bar{\mathbf{w}}}\mathcal{L}_{\mathcal{T}_i}(\mathcal{M}(\bar{\mathbf{w}}))$ wrt. $\mathcal{Z}_i$
    Possible future parameters: $\bar{\mathbf{w}}_i' = \bar{\mathbf{w}}_i - \alpha\nabla_{\bar{\mathbf{w}}}\mathcal{L}_{\mathcal{T}_i}(\mathcal{M}(\bar{\mathbf{w}}))$
  end
  Update: $\bar{\mathbf{w}} \leftarrow \bar{\mathbf{w}} - \beta\nabla_{\bar{\mathbf{w}}}\sum\mathcal{L}_{\mathcal{T}_i}(\mathcal{M}(\bar{\mathbf{w}}_i'))$
end

Algorithm 2: MAML[2] Algorithm

5.3 Related Work

MAML received multiple developments that followed the same approach of finding a warm initialization for all the tasks coming from a certain task distribution. Subsequent works highlight some of its limitations. Authors in [3, 6] improved on MAML's need for task similarity making the meta-learning more task family robust. The sensitivity of MAML to architectural details is noted and circumvented in [8]. The need for a second order term in MAML is questioned and shown that first order approximations can give equally good results in [4]. Deeper changes to the MAML method are presented in [9] to convert it to a method of probabilistic inference. Moreover, a continual learning extension where tasks are introduced sequentially is proposed in [10, 5].

Besides most of the above-mentioned developments, some of the attention went to improving the quick adaptation stage from the learned initialization. In [11] not just the initialization, but also the update direction and the learning rate are optimized. Moreover, in [12] the learning rate for the adaptation is learned with hypergradient descent. Both methods aim to limit the adaptation steps needed.

Therefore, MAML's developments also focus on quick adaptation. However, the above-mentioned articles use the Omniglot dataset [13] as a supervised classification benchmark with MAML as the baseline and some of the works recreate the sinusoidal regression task as shown in the original MAML paper [2]. Both of these settings where MAML and its variants are tested do not have the time or memory restrictions as few-shot learning problems. Yet, quick adaptation methods are still being benchmarked with these datasets.

5.4 Experimental Setting

Throughout this work uppercase bold letters (*e.g.* $\mathbf{X}$), lowercase bold letters (*e.g.* $\mathbf{x}$), and lowercase letters (*e.g.* x) are used for matrices, vectors, and scalars respectively. Moreover, the vectors are assumed to be stored in columns. Finally, the $\mathbf{I}_D$ represents a $D \times D$ identity matrix, the $\mathbf{1}_D$ and $\mathbf{0}_D$ represents $D \times 1$ vector of ones and zeros respectively.

5.4.1 Learning Problems

In this work, two families of regression tasks are considered; a linear and a nonlinear regression task family.

Consider the conventional linear regression problem

$$y = \mathbf{x}^{\mathrm{T}}\mathbf{a} + \varepsilon, \tag{5.1}$$

where $y \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{a} \in \mathbb{R}^D$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Each realization of the slope $\mathbf{a}$ corresponds to a task $\mathcal{T}$.

For the nonlinear problem family, the regression function is defined as a weighted combination of sinusoidal functions:

$$y = \sin(\mathbf{x} + \boldsymbol{\phi})^{\mathrm{T}}\mathbf{a} + \varepsilon, \tag{5.2}$$

where $y \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{a} \in \mathbb{R}^D$ and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Assuming that each realization of scale term $\mathbf{a}$ and $\boldsymbol{\phi}$ corresponds to a task observed in the environment $\mathcal{T}$.

For both learning problems each set of observed N input ($\mathbf{x}$) and its corresponding label (y) is denoted by a dataset $\mathcal{Z} := \{(\mathbf{x}_i, y_i)\}_{i=1}^N$.

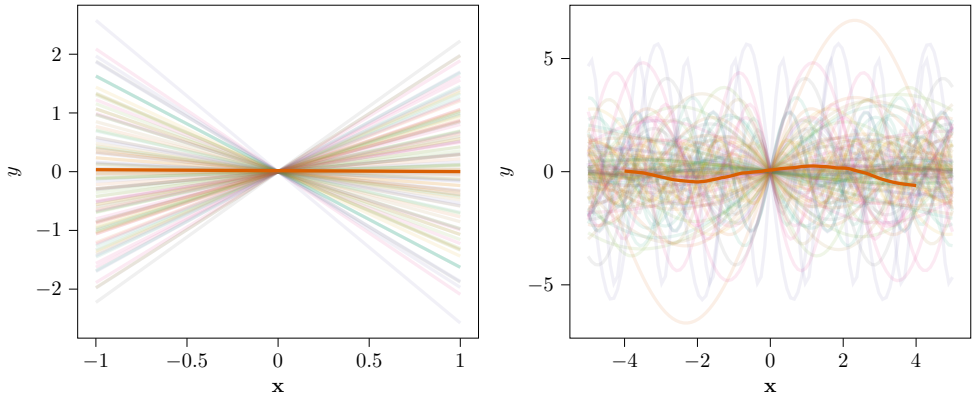
A model parameterized by $\bar{\mathbf{w}}$ is denoted by $\mathcal{M}(\mathbf{x}, \bar{\mathbf{w}}) : \mathbf{x} \to y$. A model $\mathcal{M}(\mathbf{x}, \bar{\mathbf{w}})$ that is trained with $\mathcal{Z}$ obtained from task $\mathcal{T}$ is denoted by $\hat{\mathcal{M}}(\mathbf{x})$. Noting that $\hat{\mathcal{M}}(\mathbf{x})$ for a base learner is only exposed to a single task $\mathcal{T}$ and a single dataset $\mathcal{Z}$, whereas a meta learner, in this case, MAML, are exposed to multiple tasks from $p_{\mathcal{T}}$ and multiple datasets $\mathcal{Z}$ in the meta-learning stage and then the adaptation is done as in the case of a base learner with just a single task $\mathcal{T}$ and a single dataset $\mathcal{Z}$. The discrepancy between the prediction of the estimator $\hat{\mathcal{M}}$ and y is measured in terms of squared loss $\mathcal{L} := (\hat{\mathcal{M}}(\mathbf{x}) - y)^2$. Ultimately the loss that this paper investigates is the *Expected Squared Loss* of an estimator $\hat{\mathcal{M}}$ over the $p_{\mathcal{T}}$;

$$\mathcal{E} := \iiint (\hat{\mathcal{M}}(\mathbf{x}) - y)^2 p(\mathbf{x}, y) p_{\mathcal{Z}} p_{\mathcal{T}} d\mathbf{x} dy d\mathcal{Z} d\mathcal{T}. \tag{5.3}$$

This performance measure gives rise to the *Bayes Error* to be given by σ^2 that is coming from the noise term, which represents a model that is the perfect estimator,

referred to as oracle in some of the meta-learning literature.

For all the problems the input distribution is given by $p_{\mathbf{x}} \sim \mathcal{N}(0, k\mathbf{I})$ where k is a parameter for the variance of the inputs. For the linear problem the $p_{\mathcal{T}} := p(\mathbf{a}) \sim \mathcal{N}(m\mathbf{1}_D, c\mathbf{I}_D)$ and for nonlinear problem the task distribution takes the form of a joint distribution $p_{\mathcal{T}} := p(\mathbf{a}, \boldsymbol{\phi})$ where $p_{\mathbf{a}} \sim \mathcal{N}(\mathbf{1}_D, c_1\mathbf{I}_D)$ and $p_{\boldsymbol{\phi}} \sim \mathcal{N}(\mathbf{0}_D, c_2\mathbf{I}_D)$



(a) $y = \mathbf{x}^{\mathrm{T}}\mathbf{a}$ (b) $y = \sin(\mathbf{x} + \phi)^{\mathrm{T}}\mathbf{a}$

Figure 5.1: 100 sample tasks drawn from $p_{\mathcal{T}}$ for both linear ($m = 0$ and $c = 1$) and nonlinear ($c_1 = 1$ and $c_2 = 1$) problems with low opacity and the intermediate models for MAML trained with respective $p_{\mathcal{T}}$.

5.4.2 Models

For the linear problems $\mathcal{M}(\mathbf{x}, \bar{\mathbf{w}}) := \bar{\mathbf{x}}\bar{\mathbf{w}}$ with $\bar{\mathbf{x}} \in \mathbb{R}^{1\times D+1}$ and $\bar{\mathbf{w}} \in \mathbb{R}^{D+1\times 1}$ where $\bar{\mathbf{w}} := [\mathbf{w}, b]^{\mathrm{T}}$ and $\bar{\mathbf{x}} := [\mathbf{x}, 1]$ is utilized. The optimum parameters ($\hat{\bar{\mathbf{w}}}$) for different linear models are obtained as follows:

Linear Estimator is given by the least-squares solution, $\hat{\bar{\mathbf{w}}} := (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$, where $\mathbf{X} \in \mathbb{R}^{N\times D}$ is the design matrix where the observed input data is stored in rows.

Ridge Estimator is given by $\hat{\bar{\mathbf{w}}} := (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I}_D)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$ which is obtained by minimizing the squared loss with the additional term of $\lambda||\bar{\mathbf{w}}||_2^2$.

Kernel Ridge Estimator is given by $\hat{\bar{\mathbf{w}}} = \mathbf{X}^{\mathrm{T}}\boldsymbol{\alpha}$ where $\boldsymbol{\alpha} := (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}y$ where $\mathbf{K} \in \mathbb{R}^{N\times N}$ is the *Gram Matrix* obtained by replacing $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ inner product by a kernel $\kappa(\mathbf{X}, \mathbf{X})$. Then, the prediction of the estimator takes the form $\hat{\mathcal{M}}(\mathbf{x}^*, \bar{\mathbf{w}}) = \boldsymbol{\alpha}^{\mathrm{T}}\kappa(\mathbf{x}^*, \mathbf{X})$ where $\mathbf{x}^* \in \mathbb{R}^{D\times 1}$.

For both linear and nonlinear models, gradient descent can be utilized to update the parameters of a model $\mathcal{M}$. Then,

Gradient Descent for any given model $\mathcal{M}(\mathbf{x}, \bar{\mathbf{w}})$ and a given number of iterations n_{iter} the gradient descent estimator is given by $\{\bar{\mathbf{w}}_{j+1} = \bar{\mathbf{w}}_j - \eta \nabla_{\bar{\mathbf{w}}_j} \sum_i^N (\mathcal{M}(\mathbf{x}_i, \bar{\mathbf{w}}_j) - y_i)^2\}_{j=0}^{n_{iter}}$. In other words for any given value of $\bar{\mathbf{w}}$ one gradient update is given by the gradient with respect to $\bar{\mathbf{w}}$ with a scaling parameter η.

All the models investigated can be divided into two sub-categories the models that have information regarding the task space and the ones that have not. The labels used in Section 5.5 of the models that have information regarding the task space are as follows:

- **MAML** (for linear problem): corresponds to gradient descent with n_{iter} with the adjustable parameters obtained from the mean of the tasks $\mathbb{E}[p_{\mathcal{T}}]$ with small perturbation $\delta \sim \mathcal{N}(\mathbf{0}, 0.1\mathbf{I})$ since the MAML procedure for a linear model goes to the mean of the task distribution.

- **MAML** (for nonlinear problem): an intermediate model trained with the network and meta-learning procedure given in [2] for the sinusoidal regression problem. After which gradient descent with n_{iter} is used for adaptation to a certain task.

Finally, the following model labels have information from a single task:

- **Linear**: standard least squares solution.

- **Ridge**: standard least squares solution with $L_2 - regularization$.

- **random GD**: gradient descent with n_{iter} with the adjustable parameters starting from random initialization.

- **Kernel Ridge**: kernelized (with Radial Basis Function Kernel)

For the linear problem setting, it should be noted that the optimum can always be reached when the gradient descent is utilized with an infinitely small learning rate and an infinite number of gradient steps. Thus, allowing us to investigate the difference between taking limited steps or allowing the model to go towards the optimum directly.

It should be noted that the hyper-parameters of the utilized models if there are any, are selected with a simple grid search with 20 different values, and only the one with the lowest mean expected error over the parameter under investigation are presented in the results.

5.5 Results and Discussion

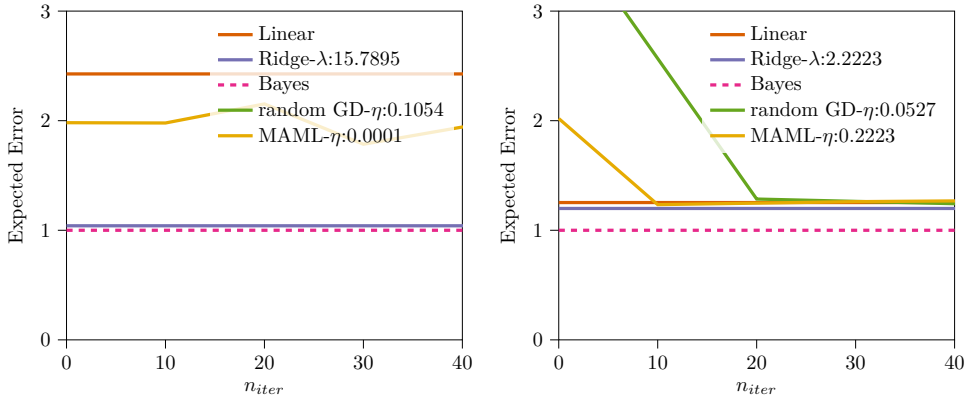
This section is dedicated to the expected performance results of a meta-learning model after adaptation and conventional base learners (*e.g.* Linear, Ridge, and Kernel Ridge Regression models), to see their performance differences under certain scenarios induced by the experimental assumptions (*e.g.* task variance, input variance, noise, and dimensionality).

Linear Problem

The linear problem introduced in Section 5.4 is characterized by the dimensionality D, number of training samples N, number of gradient steps n_{iter}, the task variance c and task mean m, and the variance of the input samples k. For the sake of brevity, only some of the parameters are discussed in this section. Unless the parameter in the configuration is under investigation, the default values are utilized. And, the default values for the experimentation $\sigma = 1$, $m = 0$, $k = 1$, $c = 1$, $n_{iter} = 1$. Moreover, the number of tasks drawn ($N_{\mathcal{T}}$), and dataset draws ($N_{\mathcal{Z}}$) for approximating the expected error given in Equation 5.3 are taken to be 100 each. Finally, the test set size is taken as 1000.

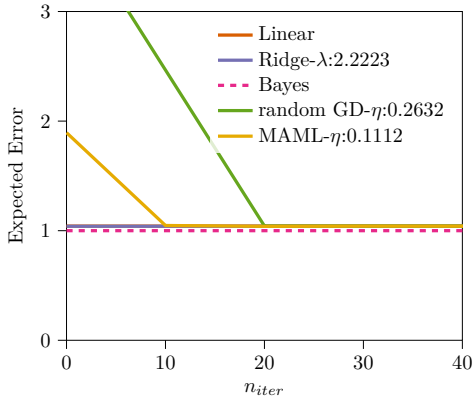
Effect of Number of Gradient Steps n_{iter}: It can be observed from Figure 5.2a that for a low number of training samples the gradient steps taken have little to no influence. But as the number of training samples increases for a given problem dimensionality the effect n_{iter} on the expected error gets much more prominent. It is evident that MAML decreases the number of gradient steps needed for convergence. Moreover, for $D = N$ case it even improves generalization after convergence too (see Figures 5.2d and 5.2a). Overall, it can be observed that the increasing n_{iter} converges towards the Ridge model variants with the exception of the $D = 1$ and $N = 1$ cases.

Effect of the Number of Training Samples N: Results of this experiment can be found in Figure 5.3 for increasing problem dimensionality. It can be seen that the Linear model suffers from singularities. For instance, in Figure 5.3b when the number of samples equals dimensionality $N = D$. However, it is able to have comparable error over all the selected problem dimensionalities. For the increasing training samples case, the Ridge model performs much better as all the cases converge towards the Bayes error. However, MAML is unable to converge towards the Bayes error. This can be attributed to the regularizing effect of the limited gradient steps (n_{iter}) allowed for the models. Overall, the improvement that the additional task-related information brings to the gradient-based models is not visible, as the random GD model is orders of magnitude higher in expected error. It is evident that although task information

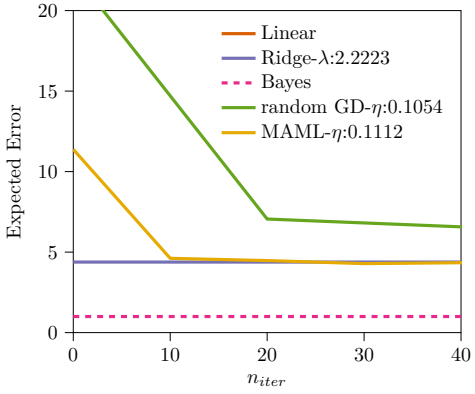


(a) $D = 1, N = 1$

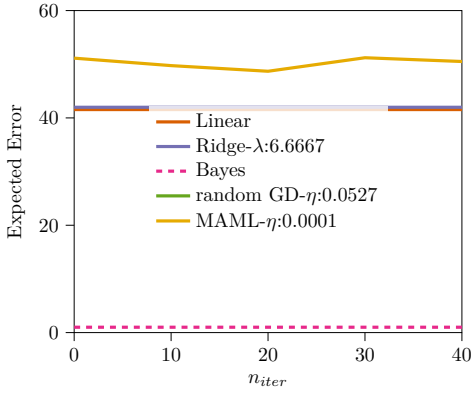
(b) $D = 1, N = 10$



(c) $D = 1, N = 50$



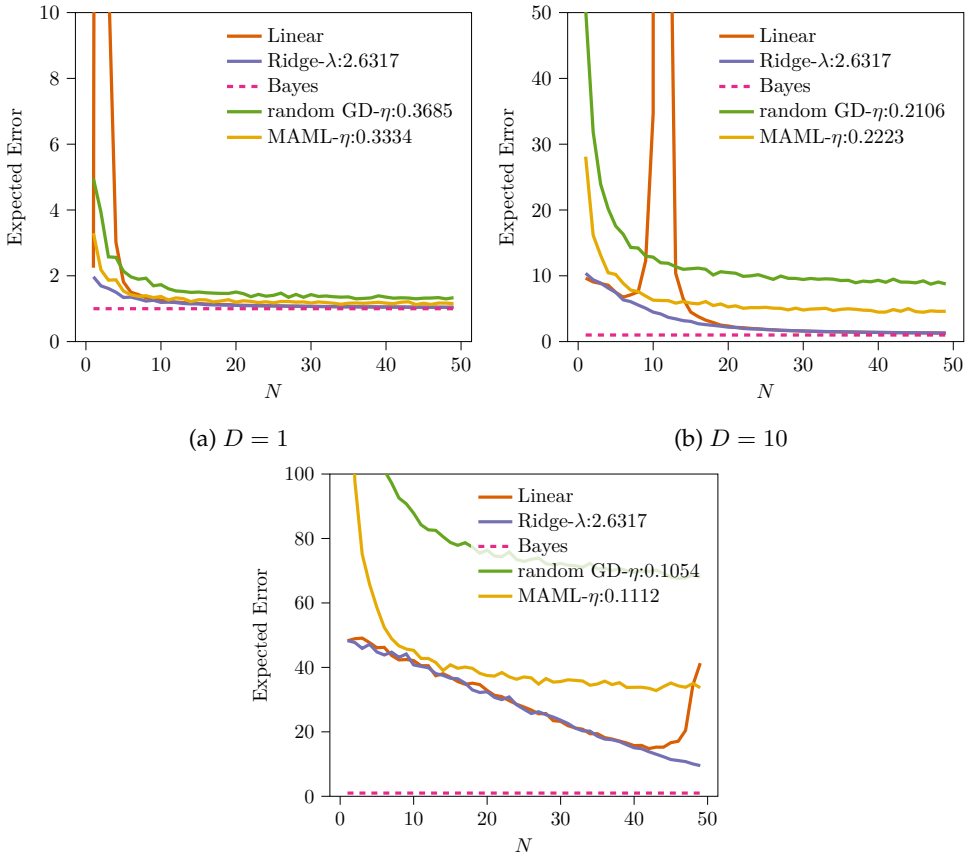
(d) $D = 10, N = 10$



(e) $D = 50, N = 10$

Figure 5.2: The expected error for the increasing number of gradient steps n_{iter} used for adaptation when changing the number of training samples for various problems of different dimensions.

provides gain over random initialization, the expected performance is hindered for the gradient-based models.



(a) $D = 1$

(b) $D = 10$

(c) $D = 50$

Figure 5.3: The expected error for the increasing number of training samples and problem dimensionality.

Effect of Task Variance c: The results of increasing task variance for various problem dimensions and various numbers of training samples can be found in Figure 5.4. The most obvious observation is that for all the models that utilize gradient descent, expected error increases, whereas the Linear model and the Ridge model are only affected by this phenomenon for problem dimensionality $D \geq N$.

In light of this observation, another important result is that for $N \geq D$ and for small task variance the gradient descent variants (except the randomly initialized model) have lower expected error than the Ridge model. However, this performance

diminishes with increasing problem dimensionality and the increasing number of training points. It is interesting to see a better performance from just one gradient step.

That is why an additional mini-experimentation is done for the GD and MAML models to investigate if there is an improvement with the additional gradient steps. The results of this experimentation are given in Table 5.1. It is observed from the table that there exists a point at which the gradient steps are hurting the expected performance one would get in this range after the second gradient step. Then, it can be conjectured that the number of gradient steps has a regularizing effect on the task distributions with small variance. Despite, the surprising results of MAML-like algorithms, the Ridge model is much more stable and performs better than gradient-based methods for $N \geq D$.

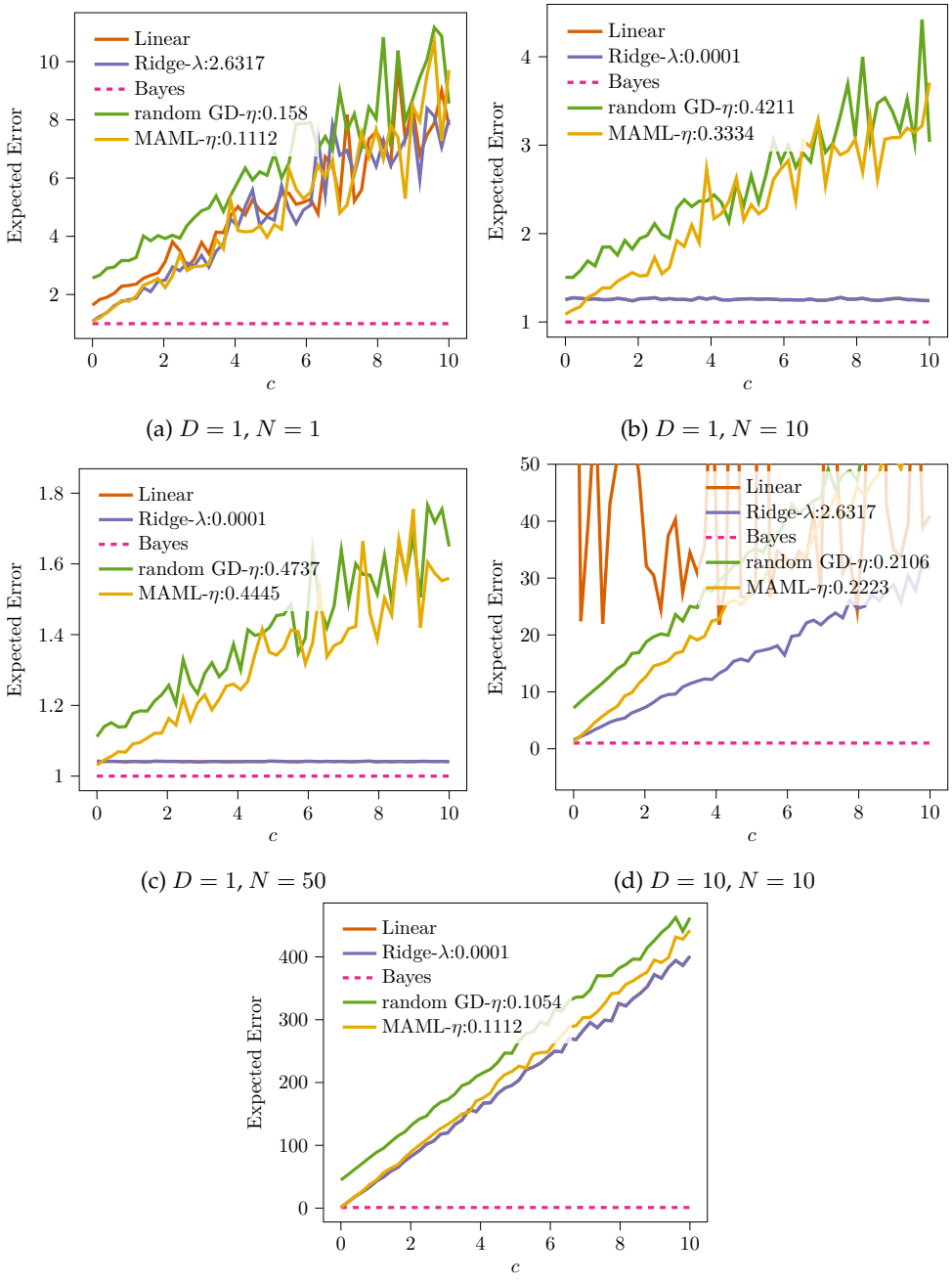
Table 5.1: Mean expected error over $c \in [0, 1]$ for different numbers of gradient steps (n_{iter}) with learning rate $\eta = 0.3234$. Results shown for $D = 1$, $N = 10$ (see Figure 5.4b).

Number of gradient steps (n_{iter})									
1	2	3	4	5	6	7	8	9	10
1.2132	**1.1938**	1.2067	1.2171	1.2318	1.2476	1.2773	1.3330	1.4622	1.7556

Effect of Task Mean m: The results can be seen in Figure 5.5. The most important observation from this experimentation is that the Ridge model has an increasing expected error for the cases of $N \leq D$ cases (see Figures 5.5a, 5.5d and 5.5e) and mostly the best λ from the trials is found to be the lowest value, which makes the Ridge model behave similar to the Linear model. Furthermore, other models which have prior task information do not seem to be affected by the task mean shifting in the task space, as expected. Again, the superiority of including information from the task space is evident as the conventional regularization cannot deal with the changing task distribution mean for $N \leq D$.

Nonlinear Problem

The nonlinear problem introduced in Section 5.4 has parameters controlling the dimensionality D, number of training samples N, number of gradient steps n_{iter}, the task variances and means m_1 and m_2, c_1 and c_2, and the variance of the input samples k. Note that only some of the parameters are discussed in this section. Unless the parameter in the configuration is under investigation, the default values are utilized. And, the default values are given as $\sigma = 1$, $m_1 = 1$, $m_2 = 0$, $c_1 = 2$, $c_2 = 2$,



(a) $D = 1, N = 1$

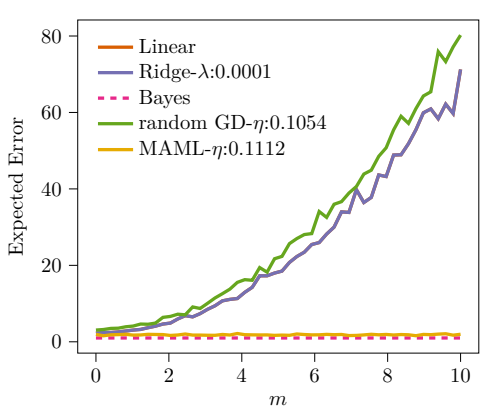
(b) $D = 1, N = 10$

(c) $D = 1, N = 50$

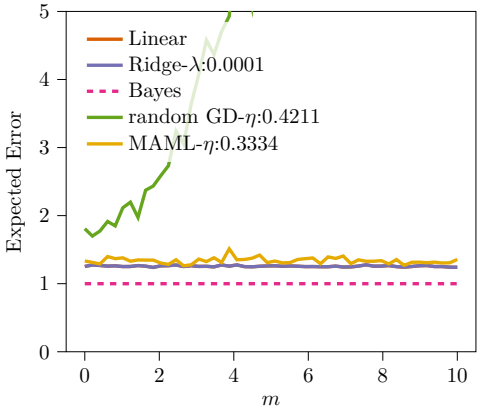
(d) $D = 10, N = 10$

(e) $D = 50, N = 10$

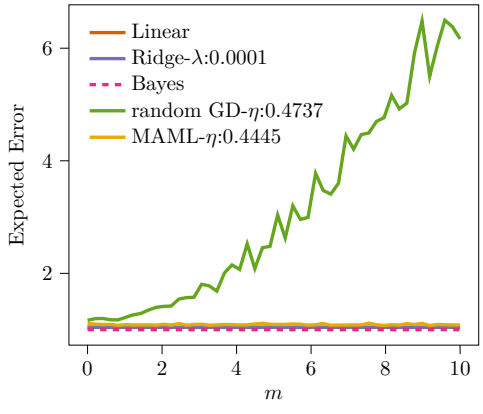
Figure 5.4: The expected error for increasing task variance c when changing the number of training samples for various problems of different dimensions.



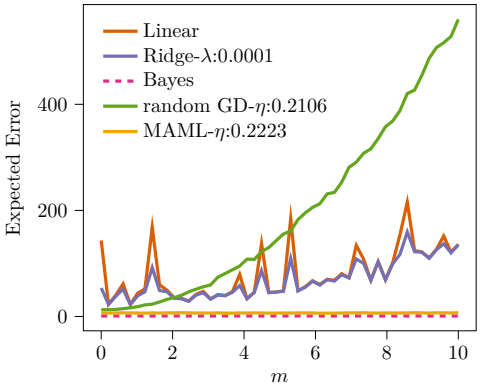
(a) $D = 1$, $N = 1$



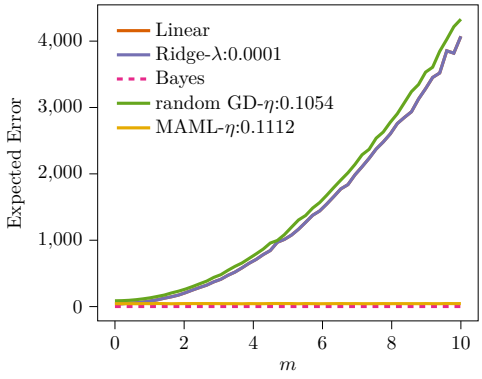
(b) $D = 1$, $N = 10$



(c) $D = 1$, $N = 50$



(d) $D = 10$, $N = 10$



(e) $D = 50$, $N = 10$

Figure 5.5: The expected error for increasing task mean m when changing the number of training samples for various problems of different dimensions.

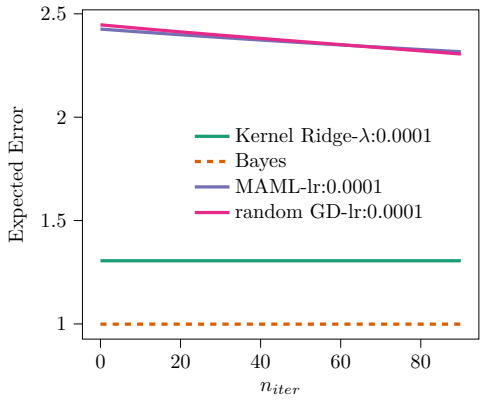
$k = 1$, $n_{iter} = 10$. Moreover, the number of tasks drawn ($N_{\mathcal{T}}$), and dataset draws ($N_{\mathcal{Z}}$) for approximating the expected error given in Equation 5.3 are taken to be 50 each. Finally, the test set size is taken as 1000.

Effect of Number of Gradient Steps n_{iter}: It can be seen from Figure 5.6a that a single realization of the Kernel Ridge model can have a lower expected error for an extreme value of 1 training sample for the 1-dimensional problem. However, as the number of training samples increases for a given problem dimensionality MAML model starts showing a lower expected error (see Figure 5.6b). Moreover, the further increase in training samples to 50 lowers the difference in expected error for given models. Another interesting observation is, that for $N \leq D$ Kernel Ridge model can achieve a lower expected error, and for all the other cases one might find a better MAML model given that sufficient gradient steps are allowed. Moreover, it can be observed that the difference between random GD and MAML is low in terms of expected error for most of the presented problems. Finally, in cases where MAML outperforms Kernel Ridge the number of gradient steps required to get a lower expected error is low.

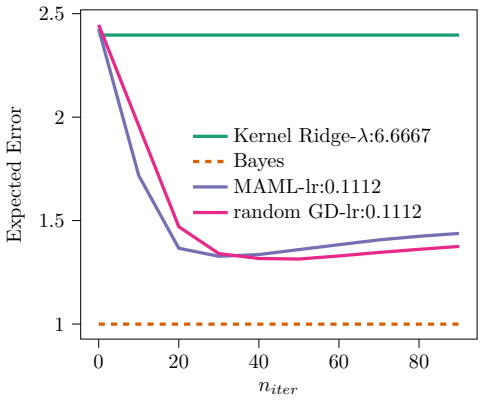
Effect of Number of Training Samples N: By looking at Figure 5.7 it can be seen that for all the given dimensionalities there exists a training sample amount where the expected error of the Kernel Ridge model is higher than MAML. Another observation is that the randomly initialized model average performance is stable over all number of training samples.

Effect of Phase Task Variance c_2: Remembering that the task variance effect for the linear problem had some interesting properties where even a single gradient step resulted in a lower expected error. One might wonder if that is the case for the nonlinear problem as well. As can be seen in Figure 5.8b a similar effect is observed for the nonlinear problem. However, for this case, the decreased expected error of MAML is only better than a randomly initialized model. This indicates that a better performance can be achieved with a regularization-based conventional learner. The only clear advantage of utilizing MAML is seen in the case where there is only 1 training sample. However, even the randomly initialized model performs better than the Kernel Ridge model for that case and what MAML adds is just a slight improvement.

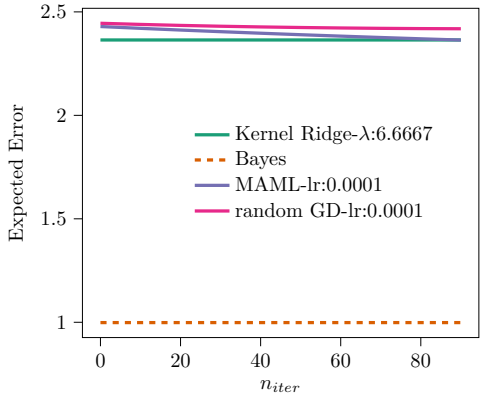
Additional experimentation results for σ for linear and nonlinear problems can be found in C, it is observed that increasing noise has similar behavior with single task learning models. Moreover, the effect of input variance is investigated and found that the gradient descent based methods perform poorer for the linear problem.



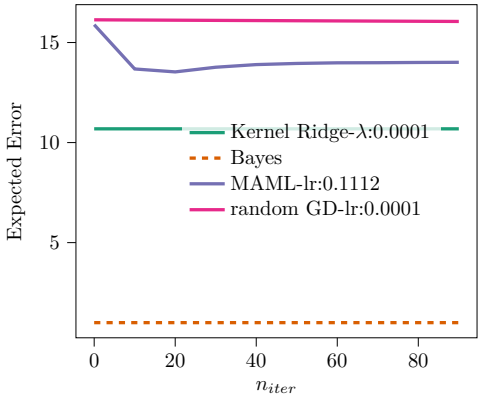
(a) $D = 1$, $N = 1$



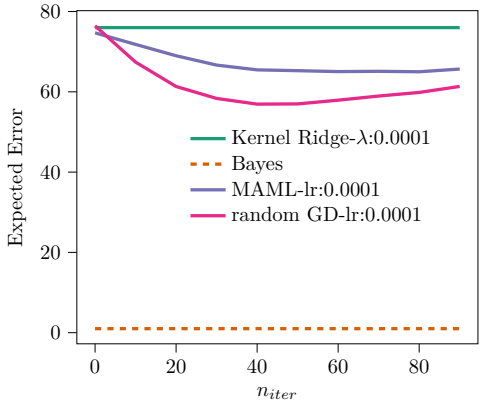
(b) $D = 1$, $N = 10$



(c) $D = 1$, $N = 50$

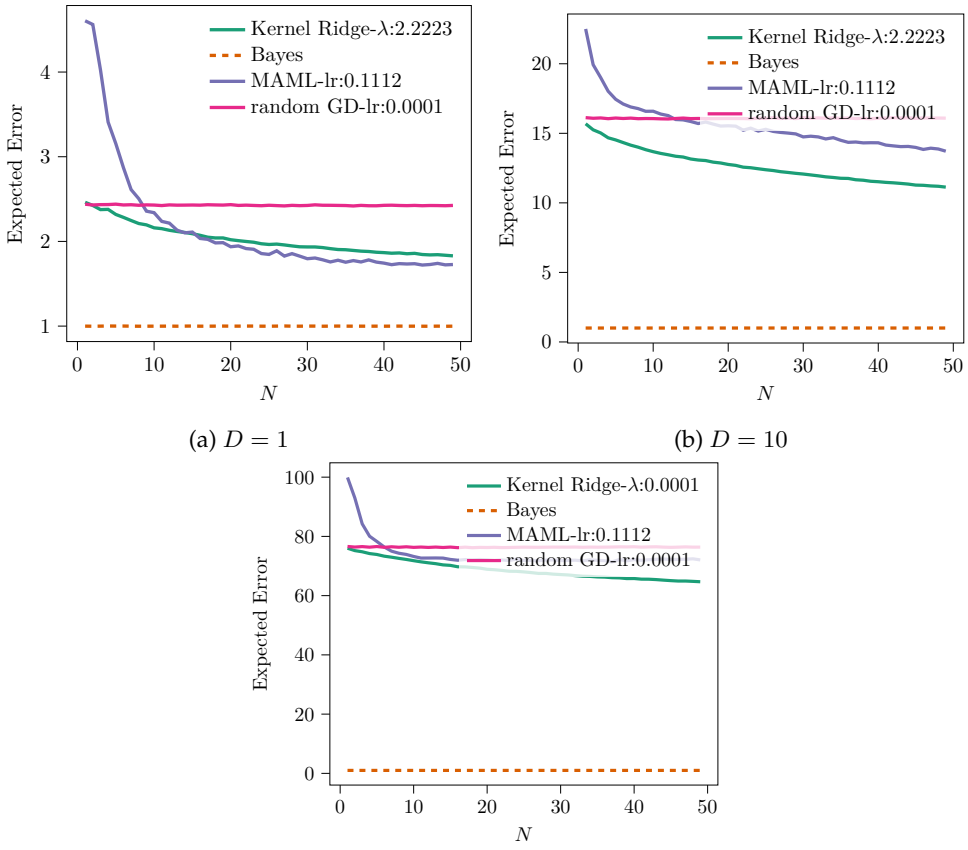


(d) $D = 10$, $N = 10$



(e) $D = 50$, $N = 10$

Figure 5.6: The expected error for the increasing number of gradient steps n_{iter} used for adaptation when changing the number of training samples for various problems of different dimensions.



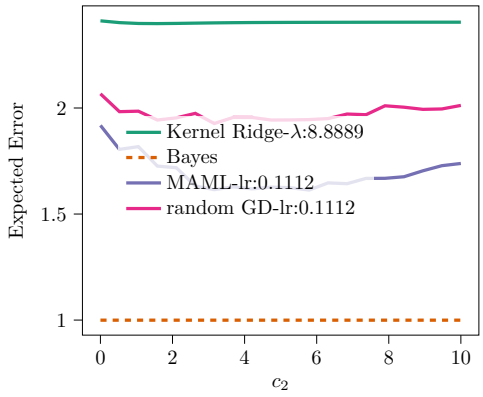
(a) $D = 1$ (b) $D = 10$

(c) $D = 50$

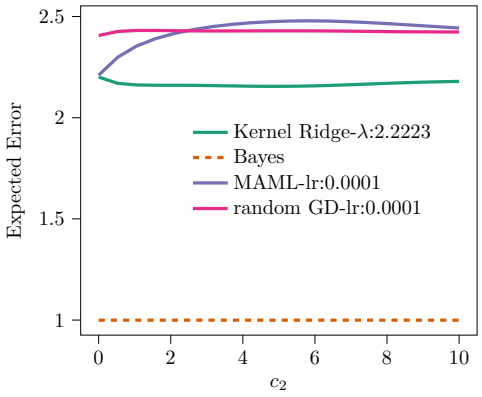
Figure 5.7: The expected error for the increasing number of training samples and problem dimensionality.

5.5.1 Discussion

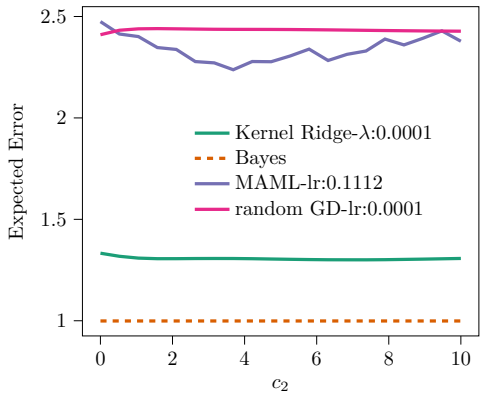
Upon our investigation, it is found empirically that meta-information about the task space can help the generalization performance in linear and nonlinear problem settings even with limited gradient steps. Increased generalization performance of MAML compared to single task learning models on expectation when the tasks that are in consideration are close to each other is observed, where the same observation is made theoretically in [14]. This observation suggests that there is a regularizing effect of limiting the gradient steps used for adaptation. We conjecture that after the meta-learning stage intermediate model parameters $\bar{\mathbf{w}}$ are closer to the test set optimum compared to the proximity of train and test set optimums. This type of behavior is investigated in [15] as well, where the large learning rate in the training phase acts as



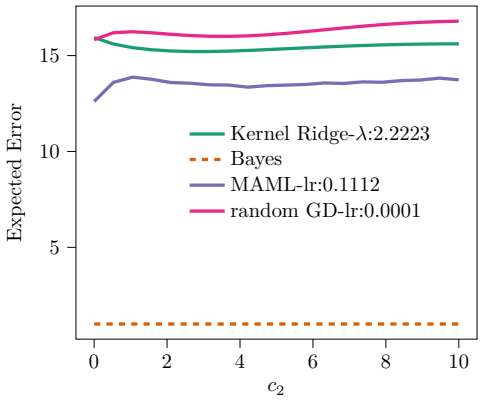
(a) $D = 1, N = 1$



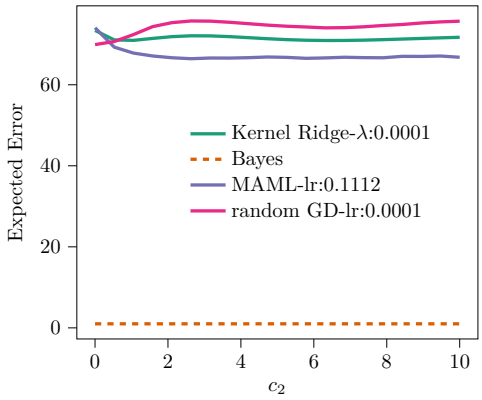
(b) $D = 1, N = 10$



(c) $D = 1, N = 50$



(d) $D = 10, N = 10$



(e) $D = 50, N = 10$

Figure 5.8: The expected error for increasing task variance for phase c_2 used for adaptation when changing the number of training samples for various problems of different dimensions.

a regularizer due to the discrepancy between train and test loss landscapes.

This limitation of adaptation steps is noted in [12, 16] that tries to improve the MAML adaptation step so that the adaptation is limited to fewer gradient steps, preferably one. Our findings suggest that the expected performance of these methods should be investigated as well since some of the generalization power of MAML might be coming from the regularization induced by not optimizing the training loss perfectly. This hypothesis is supported by the findings of [17] which concludes that the performance gain of MAML is about feature reuse instead of rapid learning.

5.6 Conclusion

Upon our investigation, we observed that single-task learners are able to compete with MAML when a limited number of gradient steps are allowed. We show that even in the case of general regularization, and when there is enough data, a single-task learner can outperform on expectation of a meta learner. Especially when the tasks observed start to deviate from each other. It indicates that the regularization-based meta-learners similar to the ones presented in [18], can be competitive and robust enough for much wider task variance. However, it also should be suitable for non-linear problem settings. Moreover, regularization-based methods similar to the ones presented in [7] for MAML can prove useful to extend the generalization performance of MAML to wider task variances.

This study only utilizes synthetic data, since the computational burden of expected error analysis is high. Thus, more in-depth expected performance should be done for the Omniglot dataset and other widely used supervised few-shot learning benchmark datasets with improvements introduced on top of MAML. We believe that understanding all the contributing factors to generalization performance is key to correct use cases of meta-learning methods.

References

[1] Sebastian Thrun and Lorien Pratt, eds. *Learning to Learn*. Boston, MA: Springer US, 1998. DOI: 10.1007/978-1-4615-5529-2.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1126–1135.

[3] Sebastian Flennerhag et al. "Transferring Knowledge across Learning Processes". In: *International Conference on Learning Representations*. 2019.

[4] Alex Nichol, Joshua Achiam, and John Schulman. "On First-Order Meta-Learning Algorithms". In: *arXiv:1803.02999 [cs]* (Oct. 2018). arXiv: 1803.02999 [cs].

[5] Jathushan Rajasegaran et al. "itaml: An incremental task-agnostic meta-learning approach". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 13588–13597.

[6] Liam Collins, Aryan Mokhtari, and Sanjay Shakkottai. "Task-robust model-agnostic meta-learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 18860–18871.

[7] Simon Guiroy, Vikas Verma, and Christopher Pal. "Towards Understanding Generalization in Gradient-Based Meta-Learning". In: *arXiv:1907.07287 [cs, stat]* (July 2019). arXiv: 1907.07287 [cs, stat].

[8] Antreas Antoniou, Harrison Edwards, and Amos Storkey. "How to train your MAML". In: *International Conference on Learning Representations*. 2019.

[9] Erin Grant et al. "Recasting Gradient-Based Meta-Learning as Hierarchical Bayes". In: *International Conference on Learning Representations*. 2018.

[10] Chelsea Finn, Kelvin Xu, and Sergey Levine. "Probabilistic model-agnostic meta-learning". In: *Advances in neural information processing systems* 31 (2018).

[11] Da Li et al. "Deeper, Broader and Artier Domain Generalization". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5543–5551. DOI: 10.1109/ICCV.2017.591.

[12] Harkirat Singh Behl, Atılım Güneş Baydin, and Philip H. S. Torr. *Alpha MAML: Adaptive Model-Agnostic Meta-Learning*. May 2019. arXiv: 1905.07435 [cs, stat].

[13] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. "The Omniglot challenge: a 3-year progress report". In: *Current Opinion in Behavioral Sciences* 29 (2019). Artificial Intelligence, pp. 97–104. DOI: https://doi.org/10.1016/j.cobeha.2019.04.007.

[14] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. "Generalization of Model-Agnostic Meta-Learning Algorithms: Recurring and Unseen Tasks". In: *arXiv:2102.03832 [cs, math, stat]* (Nov. 2021). arXiv: 2102.03832 [cs, math, stat].

[15] Preetum Nakkiran. "Learning Rate Annealing Can Provably Help Generalization, Even for Convex Problems". In: *arXiv:2005.07360 [cs, stat]* (May 2020). arXiv: 2005.07360 [cs, stat].

[16] Zhenguo Li et al. "Meta-SGD: Learning to Learn Quickly for Few-Shot Learning". In: *arXiv:1707.09835 [cs]* (Sept. 2017). arXiv: 1707.09835 [cs].

[17] Aniruddh Raghu et al. "Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML". In: *International Conference on Learning Representations*. 2020.

[18] Giulia Denevi et al. "Incremental Learning-to-Learn with Statistical Guarantees". In: *arXiv:1803.08089 [cs, stat]* (Mar. 2018). arXiv: 1803.08089 [cs, stat].

5

6 | Discussion

Evaluation can be considered one of the major enablers of a measurable progress for machine learning research [1]. The work presented in this dissertation addresses some of the main components of the evaluation topic of learning algorithms: performance measures, error estimation, statistical significance, and benchmark selection [2], through the lens of learning curves. In the following sections, some of the findings presented in this dissertation are revisited within a broader perspective.

6.1 Benchmarking

Benchmarking results on known real-world datasets are presented in Chapters 2 and 4. This is possible thanks to the online sharing of such datasets in databases, such as UCI machine learning repository [3] and OpenML [4]. This democratizes the evaluation for machine learning researchers. Although these large collections of databases contain a considerable amount of real world datasets, the general trend shows a concentrated usage of limited datasets [5]. Hence, any work that is extending our findings in terms of number of datasets is worth the effort given the potential biases that are present inherently in the datasets that were utilized.

In the remainder of the chapters, namely Chapters 3 and 5, we use synthetic datasets. There are various reasons that force people to generate and use synthetic datasets, such as privacy concerns and cost. Our reason was purely to investigate some special phenomena that may not be as easily observable in current real-world datasets. For instance, in Chapter 5 we use synthetic datasets to control the task similarities since current datasets for meta-learning applications are known to lack the variability between tasks [6].

Both real and synthetic datasets can be valuable for machine learning research, particularly in settings where data are scarce. For instance, TabPFN [7] demonstrates strong performance on real-world tabular data despite being trained exclusively on synthetic datasets. This suggests that, to some extent, synthetic data can be representative of real-world distributions. However, this approach does not fully capture real-world distributions, which is reflected in the performance gains of Real-TabPFN [8] when incorporating real-world training data during training.

These observations indicate that synthetic datasets can also play a useful role in evaluation benchmarks. Recent work such as TabArena [9] introduces the concept of a living benchmark, in which models are continuously evaluated on a curated collec-

tion of real-world datasets using a standardized pipeline. A similar approach could be adopted for synthetic datasets, enabling the community to develop a living synthetic benchmark that evolves over time or maybe even better, a combination of those can be utilized instead of relying solely on real-world or synthetic datasets. While any evaluation system, particularly those based on curated datasets, such as the one presented in [10], inevitably introduces biases. We believe that relying on combination of real-world benchmarks with synthetic ones may help cover a broader range of scenarios and yield more robust and reliable performance estimates.

6.2 Learning Curves

This thesis addresses several aspects of learning curve research, including extrapolation and the examination of key hypotheses. Based on our findings, we also identify several important topics that warrant further efforts.

Expected Performance

Every chapter of this dissertation pertains to the expected performance of machine learning models in one way or another. Focusing on expected performance is often a convenient and pragmatic choice; however it can be misleading. We showed in Chapter 2 that, for varying training set sizes, a model's generalization performance can be multimodal and skewed. Hence, the emphasis on point estimates of performance can obscure important aspects of model behavior, such as variability, robustness, and sensitivity to worst-case scenarios.

In many practical settings, understanding the reliability of a model is as critical as achieving high average performance as mentioned in Chapter 2. Therefore, while studying learning curves we might need to look at other measures that take into account the performance distribution. These distributions along learning curves can yield much more than a simple average. It can increase the quality of the model/architecture driven research. Indeed, it is rather costly maybe to obtain representative generalization performance distributions. However, given the computational power increase and potential efforts put into the computational aspects of the evaluation pipelines of learning systems, it is not impossible.

Fidelity

One important aspect that has the potential to alter the conclusions about the shapes of the learning curves is the fidelity at which the learning curves are studied. In this dissertation, fidelity refers to the resolution of the training set size used and the number of resampling per training set size. We deliberately tried to create and use high-fidelity learning curves in all of the chapters, unlike existing learning curve databases.

The motivation is straightforward: sparse learning curves, those obtained at only a

few training set sizes, can be misleading. When learning curves are sampled at coarse intervals (e.g., only at 10%, 50%, and 90% of training data), they may appear smooth and well-behaved, suggesting clear trends like monotonic improvement or smooth convexity. However, a finer-grained sampling might reveal the opposite.

As we demonstrated in Chapter 2, learning curves exhibit performance distributions with substantial outliers and heavy tails. Existing algorithms that characterize learning curve properties, monotonicity, convexity, sample complexity—rely on whatever training set resolution was used to obtain the curves. This reliance introduces systematic bias: the detected properties depend more on the sampling resolution chosen than on the underlying learning process. We hypothesise that using much denser grids would reveal even more ill-behaved curves, which is partially observed in [11].

Increasing the fidelity had several consequences, including the use of medium- to small-sized real-world datasets, as well as the consideration of a limited number of training set sizes and learning algorithms in some studies. Therefore, future work should focus on increasing dataset sizes and investigating a broader range of learning algorithms, both of which would enhance the practical relevance of learning curve research.

Hyper-parameter Optimization

A current trend in the learning curve databases created for analysis is to study a learning algorithm with fixed hyper-parameters [11, 12]. We believe that this is one of the main limitations of current learning curve research. Hence, in Chapter 2 we also investigated hyper-parameter optimization. However, it is not be feasible to thoroughly search the hyper-parameter space, especially for high-fidelity learning curves. In our work we employed grid search, which is also not scalable to many hyper-parameters. Hence, more scalable methods such as random search or evolution-based methods [6] can be used to circumvent this bottleneck and create learning curves with more complex optimization loops.

The software package for learning curve creation presented in Chapter D introduces an easy interface to enable this with minimal effort. Observing more algorithms that involve additional optimization steps can lead to changes in some of the observations obtained in this work and other learning curve research [11, 12]. Some of the general observations about the shape of learning curves given in [13] may change upon the addition of hyper-parameter optimization in the training phase. Moreover, this can also increase the relevance of learning curve research for practitioners.

Parametric Models

In Chapter 3, we proposed a data-driven method for learning curve extrapolation using a learning curve database. The motivation was the inability of many existing parametric models to extrapolate non-monotonic learning curves well. Our approach leverages a database of observed learning curves to construct a non-parametric learning curve by matching and aggregating curves with similar early-stage behavior.

Although the existing parametric methods provided in [13] provide information about the shapes of learning curves, they rely on assumptions that can be violated for certain data distributions. One way to probe the shapes of learning curves is through data-driven analysis of learning curve databases. This may help derive explicit parametric models that capture non-monotonic patterns as well. One obvious approach is to employ symbolic regression on the learning curve databases created in this work. With a restricted set of candidate functions, for example, polynomials, exponentials, and rational terms, we can obtain parametric models that are far more expressive than the current parametric models. The resulting analytical expressions can provide interpretable models of learning dynamics, enable quantitative comparison of curve shapes across tasks and models, and allow us to test hypotheses about which functional components correspond to specific training phenomena.

These parametric forms can also be incorporated for extrapolation. For instance, our proposed method in Chapter 3 can use a collection of such parametric forms and the dependence on the learning curve database can be eliminated. Or, they can be used individually for learning curve fitting and extrapolation. However, these new parametric forms may be difficult to fit, which is a known issue [13]. Therefore, improving optimization methods for fitting learning curves keeps on being an important direction for future research.

Monotonicity

Perhaps we will see the days when a learner that is robust enough to any type of data distribution and any type of evaluation metric will be provably monotone. But it is not this day, unfortunately. Currently, we only identify certain settings in which certain learners are guaranteed to improve with additional data. In Chapter 4, we present one such combination of evaluation measure and learner, where monotonicity guarantees are given in expectation for specific data distribution.

In combination with the findings of Chapter 2, the usefulness of guarantees given in expectation can be questioned. Research efforts that focus on the worst performing cases may be more meaningful than guarantees in expectation, as they provide insight into the least favourable scenarios and are therefore safer from a reliability perspective. Such guarantees may be obtained by applying extreme value statistics to random

matrix theory in simple settings such as the one considered in Chapter 4. Although for other settings the extreme value statistics may be difficult to obtain, it remains to be important.

6.3 Open-Source Toolboxes

This dissertation would not have been possible without the support of open-source machine learning and algebraic toolboxes. Over the years, machine learning research has addressed some of the open-science challenges discussed in [14], though solving those issues may have created new ones. One concern we would like to highlight is that the open-source nature of machine learning and deep learning research can sometimes have detrimental effects on research quality and reproducibility.

The openness of the machine learning community and its resources significantly accelerates development and experimentation for researchers. However, this convenience also carries a risk: subtle implementation details and methodological decisions can remain hidden when one relies exclusively on existing frameworks. This became evident while working on the software package presented in Appendix D, where many computational details would have otherwise remained unnoticed, such as additional jitter terms for numerical stability, and decisions in extreme cases of one unique or missing labels.

Modern machine learning pipelines abstract away most underlying processes and increasingly sophisticated end-to-end pipelines are readily available for public use. We believe that this encourages treating models as black boxes. While this can reduces the burden of implementing models from scratch and can shorten the research cycle [15]. We also believe that this can also risks overemphasizing benchmarking competitions at the expense of hypothesis-driven research, reducing the quality and depth of research output and increasing the reliance on the hidden decisions, the original implementers made.

These developments can also have implications for industrial roles in which machine learning expertise is valued. The ease of applying pre-built pipelines can result in overstated claims of expertise, despite limited understanding of underlying principles. This risks diluting professional standards and blurring the distinction between tool usage and methodological mastery.

This critique should not be interpreted as opposition to open science or open-source software in general. On the contrary, the democratization of computational tools and resources brings far more benefits than drawbacks. Nevertheless, researchers should remain mindful of potential pitfalls and focus not only on how tools are applied but also on what can be observed and why.

Also extending this attitude to educational practice is important for robust scien-

tific output in the field of machine learning, especially in fields at the intersection of machine learning and domain-specific applications. In such interdisciplinary contexts, understanding the assumptions underlying machine learning methods and the implementation details of software libraries can often be overlooked due to the applied nature of these systems. However, given the rapidly evolving nature of open-source software, educators should equip students with the skills to not only use current tools but also to track and understand how these tools change over time. Students should recognize that software is not infallible: bugs exist, vulnerabilities emerge, and careful scrutiny of both tools and results remains essential.

6.4 Last Words

This dissertation is unified by the theme of learning curves and data scarcity, yet it remains diverse in its approaches. It brings together some mathematics with statistics, and a fair bit of programming within the broader field of machine learning research. Some may ask whether this represents the pinnacle of machine learning research. Perhaps not.

Nevertheless, my curiosity was drawn to multiple facets of learning curves, ranging from questions of monotonicity, assumptions, computational aspects to the challenges of extrapolation. Over the years, I have come to feel that this niche area has often been overlooked, neither widely appreciated nor deeply engaged with by many of those I encountered.

Still, learning curves remain insufficiently understood, and data scarce machine learning stands to benefit substantially from a deeper and more principled understanding of them.

References

[1] Kenneth Ward Church and Joel Hestness. "A survey of 25 years of evaluation". In: *Natural Language Engineering* 25.6 (2019), pp. 753–767. DOI: 10.1017/S1351324919000275.

[2] Nathalie Japkowicz and Zois Boukouvalas. *Machine Learning Evaluation: Towards Reliable and Responsible AI*. Cambridge University Press, 2024.

[3] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017.

[4] Bernd Bischl et al. "OpenML: Insights from 10 years and more than a thousand papers". In: *Patterns* 6.7 (2025), p. 101317. DOI: 10.1016/j.patter.2025.101317.

[5] Bernard Koch et al. "Reduced, reused and recycled: The life of a dataset in machine learning research". In: *arXiv preprint arXiv:2112.01716* (2021).

[6] Pavel Brazdil et al. *Metalearning: Applications to Automated Machine Learning and Data Mining*. Cognitive Technologies. Cham: Springer International Publishing, 2022. DOI: 10.1007/978-3-030-67024-5.

[7] Samuel Müller et al. *Transformers Can Do Bayesian Inference*. 2024. arXiv: 2112.10510 [cs.LG].

[8] Anurag Garg et al. "Real-TabPFN: Improving Tabular Foundation Models via Continued Pre-training With Real-World Data". In: *Foundation Models for Structured Data workshop at ICML*. 2025.

[9] Nick Erickson et al. *TabArena: A Living Benchmark for Machine Learning on Tabular Data*. 2025. arXiv: 2506.16791 [cs.LG].

[10] Sebastian Felix Fischer, Matthias Feurer, and Bernd Bischl. "OpenML-CTR23 – A curated tabular regression benchmarking suite". In: *AutoML Conference 2023 (Workshop)*. 2023.

[11] Cheng Yan, Felix Mohr, and Tom Viering. *LCDB 1.1: A Database Illustrating Learning Curves Are More Ill-Behaved Than Previously Thought*. 2025. arXiv: 2505.15657 [cs.LG].

[12] Felix Mohr et al. "LCDB 1.0: An Extensive Learning Curves Database for Classification Tasks". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Massih-Reza Amini et al. Vol. 13717. Cham: Springer Nature Switzerland, 2023, pp. 3–19. DOI: 10.1007/978-3-031-26419-1_1.

[13] Tom Viering and Marco Loog. *The Shape of Learning Curves: A Review*. Nov. 2022. arXiv: 2103.10948 [cs].

[14] SÃ¸ren Sonnenburg et al. “The need for open source software in machine learning”. In: *Journal of Machine Learning Research* 8.Oct (2007), pp. 2443–2466.

[15] Max Langenkamp and Daniel N Yue. “How open source machine learning software shapes ai”. In: *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*. 2022, pp. 385–395.

A | Generalization Performance Along Learning Curves

A.1 Learning Curve Database Creation

The classification datasets used in this work are presented with their corresponding OpenML-ID is given in Table A.7.

In this work in order to keep the computational burden at manageable levels we decided to tune only one hyper-parameter for every given model and conducted with a simple grid search of 20 values. For continuous hyper-parameters logarithmically spaced values between 10^{-8} and 10^{-1} and for the discrete hyper-parameters regularly spaced values between 1 to 100 used. In all our experiments we use our home-brewed learning curve generation tool in C++ [1].

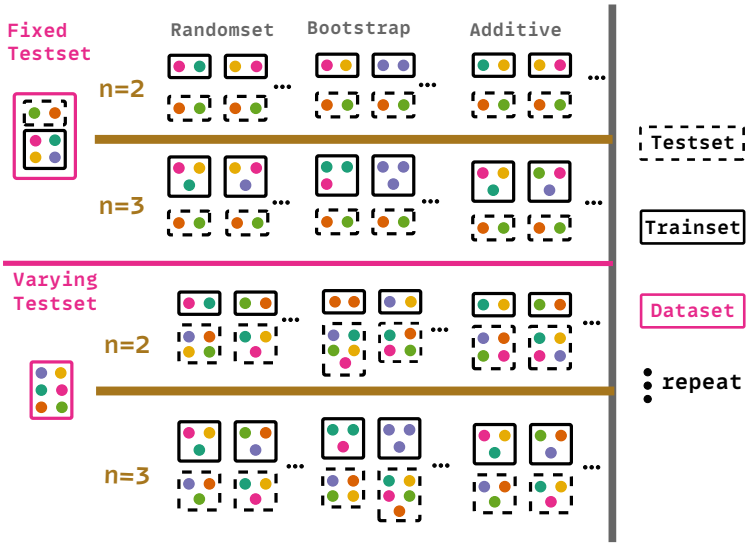


Figure A.1: Summary of sampling strategies for learning curve generation (*Additive,Bootstrap,Random Selections and with and without split*) used in this work. The dataset for training and testing for 2 repetitions is given for sample sizes 2 and 3 for both varying and fixed test set. At the top (above the pink line) a fixed test set is used, in the bottom a varying test set is used. The dashed lines encapsulate the testing set, and the solid lines encapsulate the training set used for the learning curve creation. The colored balls represent the data points in the dataset. Each ball has a unique color, hence seeing the same colored data point multiple times indicate sampling with replacement. Ellipsis represent the repetition of the experiment with different samplings for the same sample size.

In our current experiments 94 learning curves are missing out of 4800 expected learning curves. These are due to timed out experiments which are not finalized due to time constraints. The missing experiments are summarized in Tables A.1,A.2 and

A

A.3. From these one can see that the missing learning curves mostly come from Gaussian Kernel Support Vector Classifier for large datasets with hyper-parameter tuning.

Table A.1: Missing learning curves per Model

Model	Missing #
ADAB	2
GSVC	64
LREG	6
NNC	16
RFOR	6

Table A.2: Missing learning curves per Dataset ID

OpenML-ID	Missing #
1063	24
44037	20
46597	50

Table A.3: Missing Combinations for Hyper-parameter Tuning

Tuning	Missing #
Not Tuned	18
Tuned	76

A.2 Effect of Sampling Strategies

For the same set of learners used in Section 2.3.2, we examine the effect of different sampling strategies without using a separate test split, as shown in Table A.4. The only clear difference appears for the Logistic Regression model combined with Bootstrap sampling, where we observe an increase in the proportion of performance distributions classified as Gaussian. However, beyond this case, we do not observe any clear or systematic relationship between the sampling methods (Additive, Bootstrapping, and Random Selection) and the Gaussianity of the resulting performance distributions, indicating that not all models are affected by the sampling strategy in the same way.

Table A.4: Average Percentage of non-normal results for selected models for investigating the effect of sampling type.

	Additive	Bootstrap	Random
DT	97.32 ± 7.17%	99.49 ± 2.00 %	97.82 ± 5.78 %
LDC	99.63 ± 1.53%	99.87 ± 0.27 %	99.45 ± 1.52 %
LREG	98.48 ± 4.50%	97.46 ± 6.36 %	99.01 ± 3.51 %
RFOR	99.06 ± 3.90%	98.87 ± 4.20 %	98.99 ± 3.90 %

A.3 Corresponding Results for Brier and Cross-Entropy Losses Results

This section is dedicated to the corresponding result presented in Section 2.3 with Brier Loss and Cross-Entropy Loss.

Table A.5: Effect of dataset imbalance to the Gaussianity for Brier and Cross-Entropy Losses.

	Imbalanced	Balanced
Brier	95.21%	90.84%
Cross-Entropy	96.97%	94.43%

Table A.6: Average Percentage of non-normal test splits for each dataset and loss measure (based on Shapiro-Wilk test with $\alpha = 0.05$) for binary and multi-class cases of *LREG* model for Brier and Cross-Entropy Losses.

	Brier	Cross-Entropy
Binary	92.60%	99.74%
Multi-class	95.22%	99.98%

A

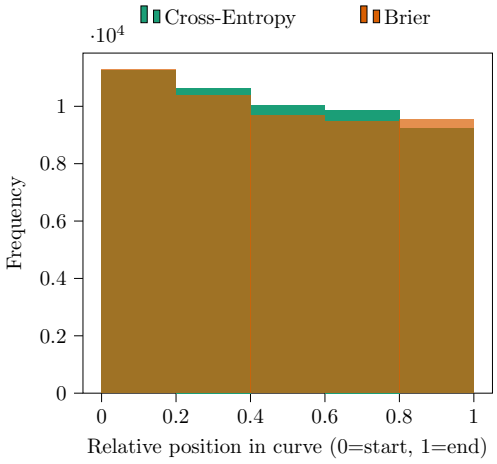


Figure A.2: Quantifying which portion of the learning curves we see more non-Gaussian performance distributions for Brier and Cross-Entropy losses.

Table A.7: Dataset information used in creating the Learning Curve Database

Property	44037	46597	46847	46733	11	37	53	61	23	1063
Number of Samples	4970	2111	383	520	628	768	270	150	1473	523
Number of Features	12	16	16	16	4	8	13	4	9	21
Number of Classes	2	7	2	2	3	2	2	3	3	2
Balanced	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗

Table A.8: Average percentage of non-normal test splits for each loss measure across datasets.

	11	**23**	**37**	**53**	**61**
Brier	97.72±7.13%	98.82±3.43%	91.42±23.83%	94.40±14.65%	95.74±16.19%
Cross-Entropy	98.99±4.17%	99.46±3.28%	93.32±22.00%	95.39±14.16%	95.13±18.01%
	1063	**44037**	**46597**	**46733**	**46847**
Brier	90.51±15.15%	82.79±33.02%	91.79±13.63%	99.11±3.87%	93.13±15.69%
Cross-Entropy	96.01±10.07%	91.86±17.80%	95.32±11.37%	99.13±3.42%	94.93±14.72%

Table A.9: Average percentage of non-normal for each loss measure across models.

	ADAB	**DT**	**GSVC**	**LDC**	**LREG**
Brier	99.29±2.95%	93.89±13.76%	96.42±16.39%	93.39±11.63%	93.14±12.24%
Cross-Entropy	99.80±1.51%	95.31±12.09%	94.97±18.92%	94.51±10.73%	99.49±2.90%
	LSVC	**NMC**	**NNC**	**QDC**	**RFOR**
Brier	91.59±27.71%	86.60±28.35%	95.07±11.91%	83.79±24.31%	92.49±15.12%
Cross-Entropy	99.99±0.07%	87.17±27.73%	94.95±12.12%	86.58±23.15%	98.33±4.34%

A

References

[1] Ozgur Taylan Turan. *Learning Curve Plus Plus*. https://github.com/taylanot/lcpp. 2024.

A

B | Learning Learning Curves

B.1 Learning Curve Database Details

In this work, both classification and regression problems are used to create a learning curve database. We introduced variety into the learning curves by altering dataset parameters and model hyperparameters. We obtain our learning curves by using 20 different hyper-parameter and 20 different dataset realizations by regular sampling from the given ranges summarized in Tables B.2 and B.3. Figures B.1 and B.2 presents one realization for some of the datasets used. The combinations of models and the datasets used for our learning curve database can be seen in Table B.1.

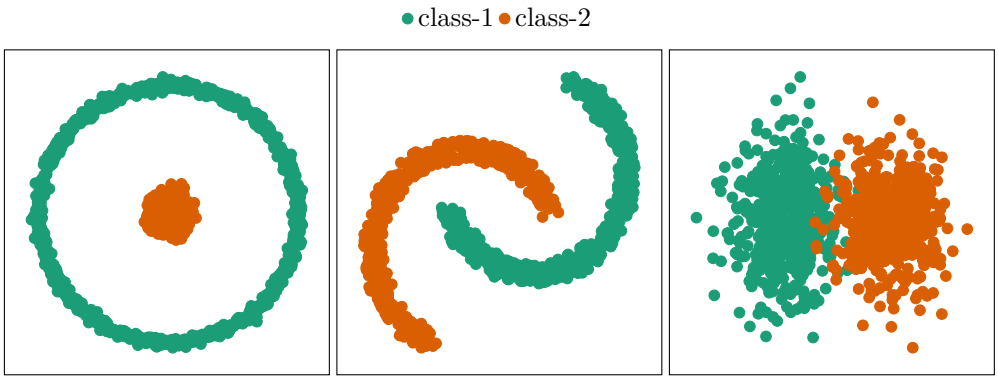


Figure B.1: Scatter plots for one realization of classification datasets (dipping, banana and Gaussian datasets from left to right).

	GAU	*BAN*	*RDIP*	*DDIP*	*ESN*	*ESC*	*ELN*	*SAW*	*DGP*
QDC	✓	✓	✓	✓					
LDC	✓	✓	✓	✓					
NNC	✓	✓	✓	✓					
NMC	✓	✓	✓	✓					
ANN					✓	✓	✓		
LR					✓	✓	✓	✓	
KR					✓	✓	✓		
GP									✓

Table B.1: Combinations of datasets and models to create learning curve database.

B

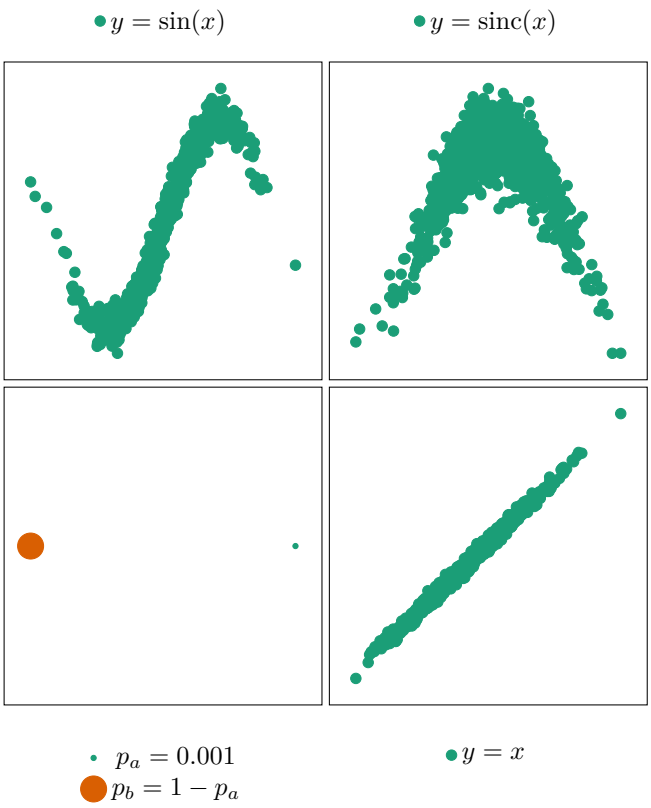


Figure B.2: Scatter plots for one realization of regression datasets (sine, sinc, sawing and linear datasets from left to right).

B.2 Computational Details

All of the learning curve generation and majority of experimentation is done in a home-brewed C++ machine learning library that can be found in `https://github.com/taylanot/mlcxx.git`. If a model that we mention was not available in *mlpack* [1] we code it from scratch (*NMC*, *LDC*, *QDC*, *KR*, *DGP*). All the datasets used for the database are created by us. *BNSL* [2] and *MDS* [3] are implemented in a library designed specifically for fitting learning curves in the python environment and is accessible at `https://github.com/taylanot/learningcurvefitting.git`. The learning curve database that we created and the experimental results can be downloaded from `https://surfdrive.surf.nl/files/index.php/s/6K4FiCtxeEdvQdx`.

Classification

Name	*Description*
Gaussian (*GAU*)	Artificial 2-class classification problem where both classes are observed from unit multivariate normal $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Means of the classes are separated from each other with $p \in [0.1, 5]$.
Banana (*BAN*)	Artificial 2-class classification problem where both classes are observed from mirrored banana shapes. Centers of the two banana shapes are separated from each other with $p \in [0.1, 5]$.
Dipping (*RDIP-DDIP*)	Artificial 2-class classification problem where first class is observed from the unit multivariate normal $\mathcal{N}(0, \mathbf{I})$ and the other class is obtained from a hyper-sphere around the first class with some additional Gaussian noise. [4]. This curve is parametrized by the dimensionality $D \in [2, 20]$ and radius of the outer hyper-spherical class $r \in [1, 100]$. See Figure B.1.

Regression

Name	*Description*
Linear (*ELN*)	$y = x + \epsilon$ where $x \sim \mathcal{N}(0, 1)$ and $\epsilon \sim \mathcal{N}(0, r)$ with $r \in [0, 1]$.
Sine (*ESN*)	$y = \sin(x) + \epsilon$ where $x \sim \mathcal{N}(0, 1)$ and $\epsilon \sim \mathcal{N}(0, r)$ with $r \in [0, 1]$.
Sinc (*ESC*)	$y = \sin(x)/x + \epsilon$ where $x \sim \mathcal{N}(0, 1)$ and $\epsilon \sim \mathcal{N}(0, r)$ with $r \in [0, 1]$.
Sawing (*SAW*)	Point masses at $(x_a, y_a) = (1, 1)$ and $(x_b, y_b) = (0.1, 1)$ with probabilities $p_a = 0.001$ and $p_b = 1 - p_a$ respectively. See Figure B.2.
Gaussian Process (*DGP*)	special dataset obtained from the prior of a 20 dimensional Gaussian [5].

Table B.2: Datasets used for the learning curve database.

Classification	
Name	*Hyperparameters*
Nearest Mean (*NMC*)	no hyperparameters
Nearest Neighbor (*NNC*)	number of neighbours: $\lambda_{\text{nn}} \in [1, 20]$.
Linear Discriminant (*LDC*)	regularization parameter for the covariance matrix: $\lambda_{\text{ld}} \in [10^{-5}, 1]$.
Quadratic Discriminant (*QDC*)	regularization parameter for the covariance matrix: $\lambda_{\text{qd}} \in [10^{-5}, 1]$.
Regression	
Name	*Hyperparameters*
Linear Ridge (*LR*)	regularization parameter: $\lambda_{\text{r}} \in [10^{-5}, 1]$.
Kernel Ridge (*KR*)	regularization parameter: $\lambda_{\text{kr}} \in [10^{-5}, 1]$ Gaussian and Laplace kernels with default length-scale $\gamma = 1$.
Gaussian Process (*GP*)	regularization parameter: $\lambda_{\text{kr}} \in [10^{-5}, 1]$ Gaussian kernel with default length-scale $\gamma = 1$.
Artificial Neural Network (*ANN*)	Adam [6] optimizer with learning rate 0.001 and width of the 2 hidden layer network changing between $[5, 25]$.

Table B.3: Models and hyperparameters, that are used for the learning curve database.

References

[1] Ryan R. Curtin et al. “mlpack 4: a fast, header-only C++ machine learning library”. In: *Journal of Open Source Software* 8.82 (Feb. 2023), p. 5026. DOI: 10.21105/joss.05026.

[2] Ethan Caballero et al. “Broken Neural Scaling Laws”. In: *The Eleventh International Conference on Learning Representations*. 2023.

[3] Rui Leite and Pavel Brazdil. “Predicting relative performance of classifiers from samples”. en. In: *Proceedings of the 22nd international conference on Machine learning - ICML ’05*. Bonn, Germany: ACM Press, 2005, pp. 497–503. DOI: 10.1145/1102351.1102414.

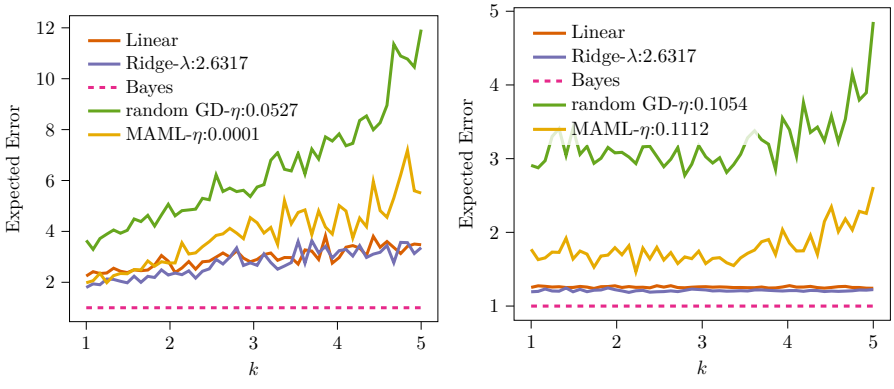
[4] Marco Loog and Robert P. W. Duin. “The Dipping Phenomenon”. In: *Structural, Syntactic, and Statistical Pattern Recognition*. Ed. by Georgy Gimel’farb et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 310–317.

[5] Peter Sollich. *Gaussian Process Regression with Mismatched Models*. 2001. arXiv: cond-mat/0106475 [cond-mat.dis-nn].

[6] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

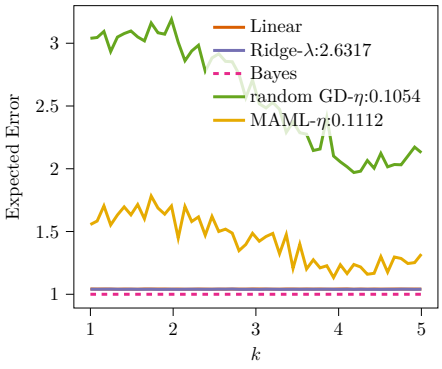
B

C | When MAML Learns Quickly Does it Generalize Well?

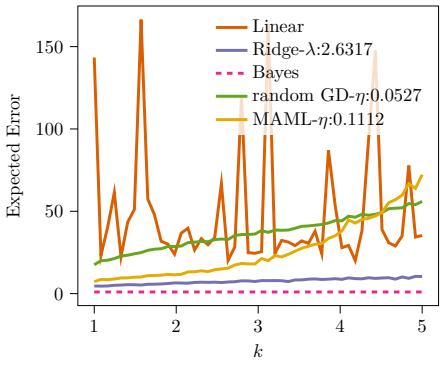


(a) $D = 1, N = 1$

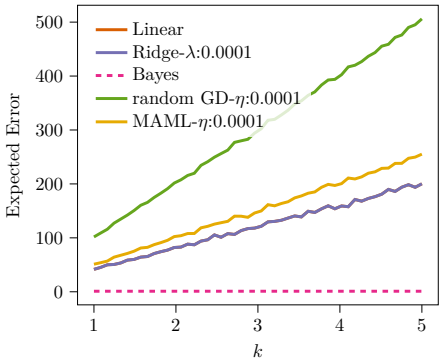
(b) $D = 1, N = 10$



(c) $D = 1, N = 50$



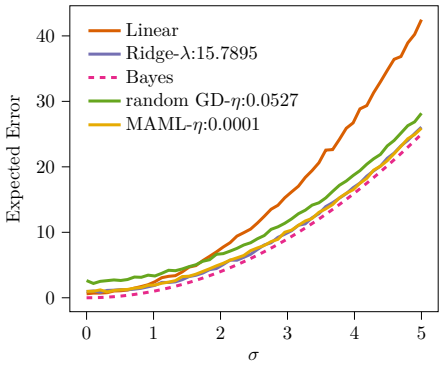
(d) $D = 10, N = 10$



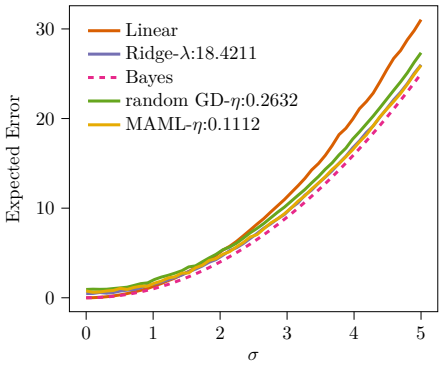
(e) $D = 50, N = 10$

Figure C.1: **[Linear Problem]**: The expected error for increasing input variance k when changing the number of training samples for various problems of different dimensions.

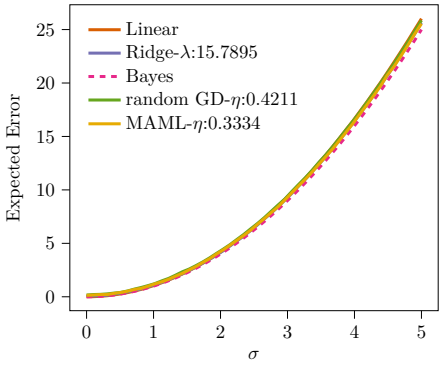
C



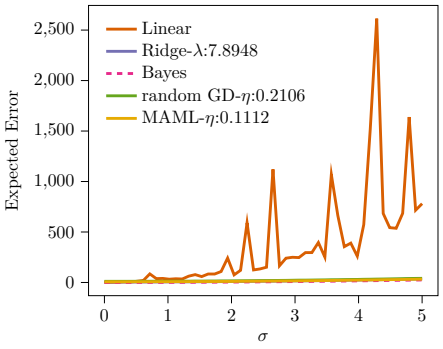
(a) $D = 1, N = 1$



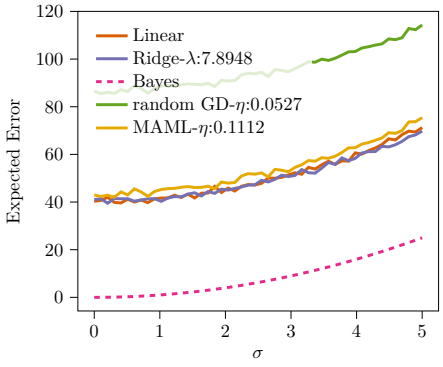
(b) $D = 1, N = 10$



(c) $D = 1, N = 50$

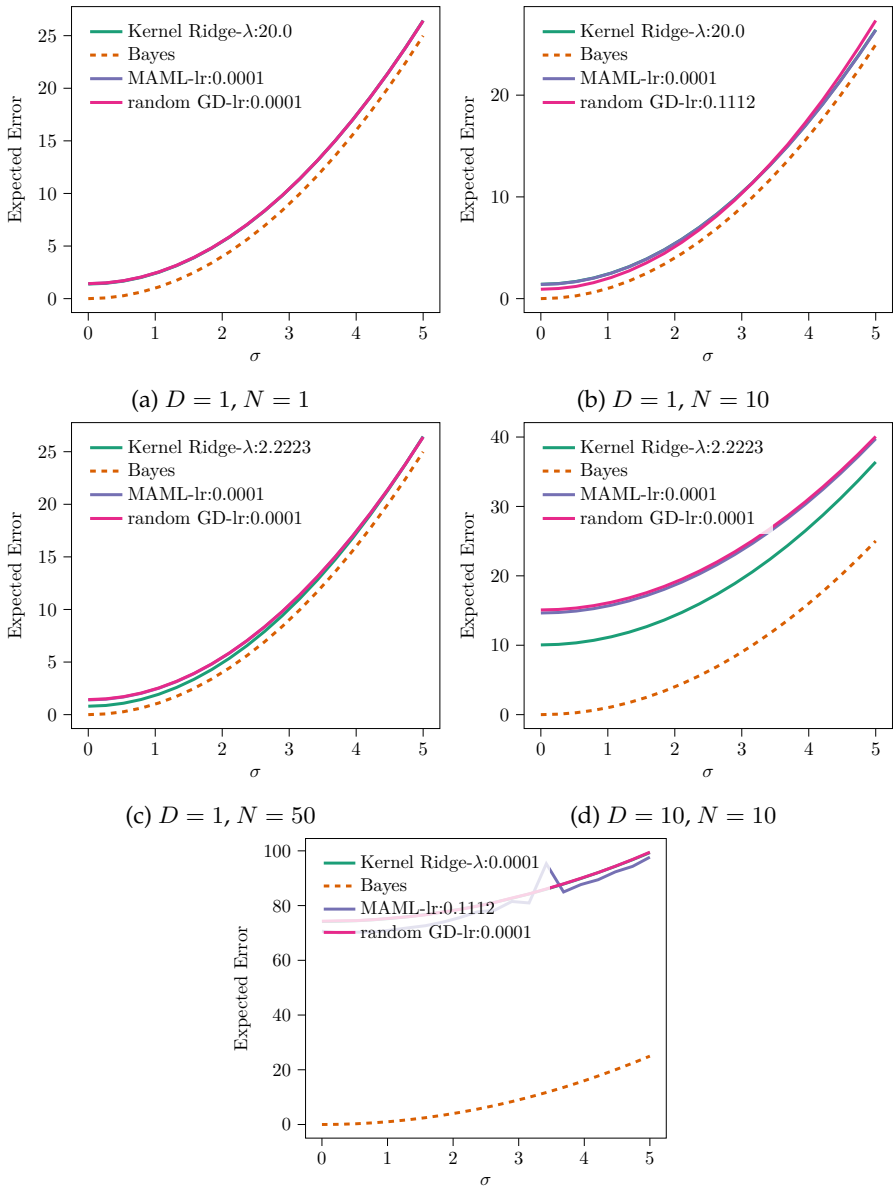


(d) $D = 10, N = 10$



(e) $D = 50, N = 10$

Figure C.2: **[Linear Problem]**: The expected error for increasing noise standard deviation σ when changing the number of training samples for various problems of different dimensions.



(a) $D = 1, N = 1$

(b) $D = 1, N = 10$

(c) $D = 1, N = 50$

(d) $D = 10, N = 10$

(e) $D = 50, N = 10$

Figure C.3: [**Nonlinear Problem**]: The expected error for increasing noise standard deviation σ when changing the number of training samples for various problems of different dimensions.

C

D | Learning Curve Plus Plus

A learning algorithm is said to learn if its performance on a given task improves with experience [1]. This fundamental definition links the size of training data to the generalization performance of the model. In supervised learning, a learning curve *depicts how generalization performance evolves as a function of the training set size. Collections of such data, known as* learning curve databases *track the performance of diverse machine learning algorithms (learners) across multiple tasks as they observe increasing amounts of training data.*

Learning curve databases are valuable for model selection *and for* estimating the amount of data needed *to achieve a target performance. These applications typically assume learning curves are monotonic and convex. However, findings of [2], [3] and [4] suggest that learning curves often exhibit more complex and irregular behavior. Sparse sampling of training sizes limits the ability to fully characterize these behaviors, highlighting the need for* high-fidelity learning curves *to investigate them.*

LCPP (Learning Curve Plus Plus)*is a C++ library that allows for learning curve creation of machine learning models. LCPP enables its users to obtain learning curves for variety of learners on any supervised learning dataset with/out hyper-parameter tuning, enabling model selection and training data requirement determination.*

D.1 Statement of Need

Generally, creating learning curves is computationally expensive because it requires repeatedly training algorithms on many subsets of varying training sizes. Consequently, learning curves are often computed for a limited number of training set sizes. For example, while creating learning curve databases [3] and [2] limited number of training set-sizes are investigated, moreover, these generations are done only for fixed learners without hyper-parameter tuning.

To empower the machine learning community to generate richer, more detailed learning curves, we propose LCPP, a C++ library for scalable learning curve generation. LCPP offers several features; first several approaches for splitting a given dataset into training and test sets of varying sizes (where training sets can be drawn randomly or incrementally, where test sets can be fixed or vary in size). Next, unlike most existing tools that fix hyper-parameters during learning curve creation, LCPP integrates hyperparameter optimization routines from mlpack [5], enabling optimized learner evaluations.

Published as: **O. Taylan Turan** et al. "LCPP: Learning Curve Plus Plus". In: Journal of Open Source Software 11.120 (2026).

LCPP also includes a simple dataset container for access to OpenML datasets [6], with built-in support for complete dataset transformations and train/test splits, allowing users to directly measure the generalization performance of models available in mlpack [5] and some other learning algorithms included in itself, such as kernel ridge regression, discriminant classifiers, multi-class classification extensions of binary classifiers.

D.2 State of the Field

Several tools are available in the Python ecosystem for learning curve generation. A flexible interface for constructing learning curves is provided in `scikit-learn` [7]. It allows cross-validation strategies to be combined with a learner. However, its extensibility and suitability for constructing high-fidelity learning curves remain limited. `LCDB 1.0-1.1` [2, 3] primarily serve as wrappers around existing learning curve databases. While learning curve generation is possible, it requires additional modification of the provided scripts and is not a central design focus.

In the C++ domain, we are not aware of any tool explicitly designed for learning curve generation. LCPP addresses this gap. It is modular by design, can be extended to support a wide range of learning curve research workflows, and can be deployed in high-performance computing environments with minimal overhead. In addition, similar to `mlpack` [5], it can be valuable for embedded and low-resource environments, for model-selection and hyper-parameter tuning purposes.

D.3 Software Design

LCPP is designed for easy deployment on high-performance computing (HPC) environments. With little effort it can efficiently run large-scale experiments in parallel, ensuring reproducibility and scalability. Moreover, it supports easy and light-weight check-pointing, allowing high-fidelity (both in terms of the training set size resolution and also the times the training set is resampled) learning curves to be created in multiple sessions. This structure also enables the missing experiments to be investigated easily.

It is also designed with future-proofing in mind. Adoption of the mlpack [5] conventions LCPP has access to wide range of learning algorithm access. As mlpack [5] continues to expand the number of supported models will also increases. In addition, LCPP is not restricted by this, any model that is using the same conventions as mlpack [5] and relies on armadillo [8] and ensmallen [9] can also be used without an effort.

D.4 Research Impact Statement

LCPP is used in Chapeter 2 and 3 to generate large-scale learning curve databases by considering many degrees of freedom involved in this process. By enabling tracking of the generalization performance across machine learning models, LCPP facilitates a systematic learning curve creation with a fast development and deployment cycle. We hope that it serves as a foundation for future learning curve research.

References

[1] Tom M. Mitchell. *Machine Learning*. Nachdr. McGraw-Hill Series in Computer Science. New York: McGraw-Hill, 2013.

[2] Cheng Yan, Felix Mohr, and Tom Viering. *LCDB 1.1: A Database Illustrating Learning Curves Are More Ill-Behaved Than Previously Thought*. 2025. arXiv: 2505.15657 [cs.LG].

[3] Felix Mohr et al. "LCDB 1.0: An Extensive Learning Curves Database for Classification Tasks". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Massih-Reza Amini et al. Cham: Springer Nature Switzerland, 2023, pp. 3–19.

[4] Tom Viering and Marco Loog. *The Shape of Learning Curves: A Review*. Nov. 2022. arXiv: 2103.10948 [cs].

[5] Ryan R. Curtin et al. "Mlpack 4: A Fast, Header-Only C++ Machine Learning Library". In: *Journal of Open Source Software* 8.82 (Feb. 2023), p. 5026. DOI: 10.21105/joss.05026. arXiv: 2302.00820 [cs].

[6] Bernd Bischl et al. "OpenML: Insights from 10 years and more than a thousand papers". In: *Patterns* 6.7 (2025), p. 101317. DOI: 10.1016/j.patter.2025.101317.

[7] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[8] Conrad Sanderson and Ryan Curtin. "Armadillo: A Template-Based C++ Library for Linear Algebra". In: *The Journal of Open Source Software* 1.2 (June 2016), p. 26. DOI: 10.21105/joss.00026.

[9] Shikhar Bhardwaj et al. "Ensmallen: A Flexible C++ Library for Efficient Function Optimization". In: (Dec. 2018). DOI: 10.5281/zenodo.2008650. arXiv: 1810.09361 [cs, math].

Acknowledgements

I want to keep it simple and precise so that you can easily find yourself and see what I was thankful for during this agonizing stretch. (Agonizing not because of the PhD journey itself, but because of the life that coincided with it.) I tried, to the best of my ability, to thank everyone in their own language, if I know it to some extent. I apologize in advance for sudden language transitions.

Ik wil graag beginnen met het bedanken van mijn academische familie. **Marcel**: ik ben je zeer dankbaar dat jij mij en Yuko hebt beschermd tegen alles wat er buiten gebeurde. Dat gaf mij een gevoel van veiligheid, en dat was hard nodig om deze promotie op een goede en tijdige manier af te ronden.

David en Marco, heel erg bedankt voor de vuurwerkshow tijdens onze afspraken. Het was een bijzondere omgeving voor onderzoek en ideevorming. Ik ben ontzettend dankbaar voor jullie en trots dat ik jullie student mocht zijn. Wat ik van jullie heb geleerd, zal ik mijn hele leven meenemen.

Ik ben speciaal dankbaar voor David, die me opnieuw leerde hoe ik vragen kan stellen, hoe ik kan onderwijzen en hoe ik weer kan lezen en schrijven, altijd met geduld en begrip, ongeacht de omstandigheden. En Marco, die me in de laatste fase van dit werk, terwijl we apart waren, liet zien hoe ik rechtop kan blijven staan zonder mezelf op te geven, en die me leerde dat stoppen niet altijd falen is, hoe en wanneer ik nee kan zeggen, hoe belangrijk precisie is, hoe ik om hulp kan vragen en dat ik niet zomaar dingen moet aannemen.

Nogmaals dank jullie wel, voor het laten zijn wie ik echt ben, de vrijheid om onderzoek te doen naar wat ik wil en de aanmoediging tijdens dit proces. Hopelijk hebben jullie het samenwerken ook als mooi en gezellig ervaren.

I would also like to thank Frans and Iuri. Although we were not able to collaborate as much as I had hoped, I have learned a great deal from both of you. A special thanks to Iuri for introducing me to `vim`, which made PhD life twice as fun.

Now, with that out of the way: for the rest of you who are looking for your name, simply find the group you belong to, and you will see why I am thanking you next to your name(s). If you cannot find your name, I apologize, I may have missed you unintentionally. Please, feel free to reach out to me via an email with the subject *whataboutme?*, and I will be happy to thank you as well.

Table D.1: PRB is a large group. I thank the people on the Bio side for the occasional chats and encounters in graduate school courses. To the rest of the PR people, thanks...

Wie/Who/Kime	**Voor/For/Neden**
Aleksandr	showing me how to elegantly handle an extremely delicate collaboration.
Arman	the office massages and distributing your chillness and enthusiasm for engineering.
Attila&Xin *Robert-Jan&Ombretta*	sharing your office during COVID times, the good spirits and a smiley faces without exception.
Aurora&Hesam *Chengming&Alejandro*	letting me disturb your peace and showing me that you can do tons of sports on top of a PhD.
Bernd	all the Hi's!
Cheng	suffering the learning-curve faith with me.
Chirag	being there for anyone and everyone.
Gijs	de wandelingen naar het station, gesprekken over van alles en nog wat, en alle anonieme röntgenfoto's.
Jan&Hayley&Jesse	sharing your perspectives on science and putting in the effort when you did not even have to.
Jim&Myrthe	een ander perspectief op de Nederlandse gezondheidszorg.
Mahdi&Ramin	all the poetic talks and sincere laughs and ranting.
Ojas	the technological sharing and caring.
Rickard	showing how to sail through a PhD with grace.
Ruud	mij helpen mijn eigen computer onder begeleiding te repareren en samen Nederlands oefenen zonder dat je je verveelde.
Sander	alle ritten naar Den Haag.
Thomas	de oprechte interesse in anderen, het lachen en Xiangwei niet alleen laten.
Tiffany	occasional visits and discussions about life and psychology.
Tom	DnB-kameraadschap en soms mijn onofficiële begeleider zijn.
Xiangwei	being the Sam to my Frodo.
Yavuz&Osman	eski evimi, anlık olsa da, hissettirdiğiniz için.
Yuko	making me feel less alone in the darkest corners of a PhD.
Ziqi	all the BBQs.

Table D.2: Friends outside of the work environment. Thanks...

Wie/Who/Kime	**Voor/For/Neden**
Alican	bütün çılgınlıklar için.
Ecem&Erhan *Pelin&Fatih&Cinar*	güler yüzünüzü, zorlu günlerde çevreme yaydığınız için.
Mert&Koray	bütün güzel kaçamaklar, yolculuklar, buluşmalar ve dertleşmeler için.
Oldies But Goldies	rakı sofraları için.
Sinan&Merve&Defne	abilikler, ablalıklar ve tatlışlıklar için.
Ekin&Marta	all the biers, ouzos, rakis, fish and naturalization ceramonies.
Erik&Lara&Nora	being the best neighbours, game nights, and spontaneity.
Glenn	the best hugs.
Guus&Marcel&Nick	mij leren goed te leven; met stampotjes, nachtelijke frikandellen en kroketten, pilsjes en boetes voor wildplassen.
Irene&All DnB people	het delen van alle dansvloeren en festivalterreinen.

Table D.3: Mijn familie in Nederland. Dank jullie wel...

Wie	**Voor**
Johan&Truike	jullie openden je armen voor mij alsof ik één van jullie was, in zowel makkelijke als moeilijke tijden.
Meike&Dennis	mijn eerste Nederlandse huwelijksbeleving en waardering, voor het delen van mijn muzieksmaak en het verspreiden van je rust en warmte naar mij, ook al zijn we ver van elkaar en zien we elkaar niet vaak.
Cato&Loes *Renske&David*	Het bieden van een familie in de buurt, gezellige logeerpartijtjes, lekker knutselen en pannenkoekenavonturen.
Femke&Beth	het steunen van deze gekke jongen door dik en dun, het zijn van mijn thuis en mijn rots in de branding die me ankerde in de realiteit en het leven, het vertragen van mijn leven zodat ik de mooie dingen weer zie, het laten voelen dat rust geen stilstand is maar bewust leven, en het tonen dat succes soms zo eenvoudig kan zijn als zijn wie je bent en weten waar je van houdt.

Table D.4: Türkiye'deki ailem. Teşekkürler...

Kime	**Neden**
Oktay&Bahar *Memo&Ayşe*	küçük ailemi büyüttüğünüz için.
Nahideh&Aynur	bütün sıkı kucaklaşmalar için.
Haluk&Ayşe	Avrupa'yı bana getirdiğiniz için.
Mert&İpek *Mine&Cesur*	çocukluğumdan beri evinizi, sofranızı, hayatınızı benimle paylaştığınız için.
Irmak&Hasan&Methiye	nasıl arkadaş olunur gösterdiğiniz için.
İdil&Nilgün&Ercu	bütün kahkahaları ve gözyaşlarını paylaştığınız için.
Nesrin	ikinci bir anne/baba olduğun için.
Tugay	bana hayatı nasıl sakince yaşayacağımı ve ölümü nasıl telaşsız karşılayacağımı gösterdiğin için.
Şükran	her gün bana ve etrafına yansıttığın güzellikler ve zorluklar karşısında güçlü ve pozitif kalmayı öğrettiğin için.

Curriculum Vitæ

02-01-1993 Born in Çankaya, Türkiye.

Education

2011-2017 **Middle East Technical University**, Türkiye

Bachelor of Science, Environmental Engineering (2011–2016)

Bachelor of Science, Civil Engineering (2013–2017)

2018-2026 **Delft University of Technology**, The Netherlands

Master of Science, Civil Engineering (2018–2020)

Doctor of Philosophy, Computer Science (2020–2026)
Thesis: Learning Curves with Little Data
Promoters: Prof. dr. ir. M.J.T. Reinders and Prof. dr. M. Loog

Research Output

6. **O. Taylan Turan** and David M.J. Tax. "LCPP: Learning Curve Plus Plus". In: *Journal of Open Source Software* 11.120 (2026), p. 9737. DOI: 10.21105/joss.09737

5. **O. Taylan Turan**, Marco Loog, and David M. J. Tax. "On Sample-Wise Strict Monotonicity with a Gradient Update". In: *Advances in Intelligent Data Analysis XXIV*. ed. by Mitra Baratchi, Siegfried Nijssen, and Jan N. van Rijn. Cham: Springer Nature Switzerland, 2026, pp. 72–83. DOI: 10.1007/978-3-032-23833-7_6

4. **O. Taylan Turan**, Marco Loog, and David M.J. Tax. "Generalization Performance Distributions Along Learning Curves". In: *Pattern Recognition Letters* (2026). DOI: https://doi.org/10.1016/j.patrec.2026.01.003

3. **O. Taylan Turan** et al. "Learning Learning Curves". In: *Pattern Analysis and Applications* 28.15 (2025). DOI: 10.1007/s10044-024-01394-6

2. Aleksandr Dekhovich, **O. Taylan Turan**, Jiaxiang Yi, and Miguel A. Bessa. "Cooperative data-driven modeling". In: *Computer Methods in Applied Mechanics and Engineering* 417 (2023), p. 116432. DOI: https://doi.org/10.1016/j.cma.2023.116432

1. **O. Taylan Turan**, David M.J. Tax, and Marco Loog. "When MAML Learns Quickly, Does It Generalize Well?" In: *BNAIC BeNeLearn*. 2022