Comparing Learning Curve Extrapolation Methods in different Contexts

Kielhöfer, Lionel

Comparing Learning Curve Extrapolation Methods in different Contexts

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
PHYSICS

Author :	Lionel Kielhöfer
Student ID :	s1978462
Supervisor :	Dr. J.N. van Rijn
	Dr. F. Mohr
	Dr. V. Dunjko
Second corrector :	

Leiden, The Netherlands, November 30, 2023

Comparing Learning Curve Extrapolation Methods in different Contexts

Lionel Kielhöfer

Huygens-Kamerlingh Onnes Laboratory, Leiden University
P.O. Box 9500, 2300 RA Leiden, The Netherlands

November 30, 2023

Abstract

Learning curves are important for decision making in supervised machine learning. They show how the performance of a machine learning model develops over a given resource. In this work, we consider learning curves that model the performance of a machine learning model as a function of the number of data points used for training. For decision making, it is often useful to extrapolate learning curves, which can be done, for example, by fitting a parametric model based on the observed values, or by using an extrapolation model trained on learning curves from similar datasets. We perform an analysis comparing these two techniques with different observations and prediction objectives. When only a small number of initial segments of the learning curve have been observed we find that it is better to rely on learning curves from similar datasets. Once more observations have been made, a parametric model, or just the last observation, should be used. Moreover, we find that using a parametric model is mostly useful when the exact value of the learning curve itself is of interest. Lastly, we use this knowledge to improve machine learning on a particle physics dataset.

Contents

Chapter 1

Introduction

Research into machine learning and its development is prominent in areas such as particle physics. Particle physics, which is the study of fundamental particles, has in recent years felt the effect of modern technology vastly increasing the amount of available data. It is usually inefficient to process this by standard means. Machine learning provides a solution to do this efficiently [1].

Learning curves are used for various types of decision making in supervised learning. They show how the performance of a machine learning algorithm develops over a given resource, for example the number of epochs, run time, or number of data points used for training. In this work, we look specifically at learning curves across data points. They are generally used in the following three decision making situations [21, 29]:

- Early stopping [13, 25]; for determining when training a given machine learning model on more budget would not cause significant improvement.
- Early-discarding [15, 22, 27]; this situation focuses on picking the best learning algorithm from a set of options. Instead of training all of the learning algorithms on the entire dataset, which is computationally costly, they are trained on increasing budgets. The learning algorithms that are predicted to perform the worst can then be discarded early on.
- Data-acquisition [18, 30]; in this situation the focus is on predicting if acquiring additional data would significantly increase the perfor-

mance of a machine learning algorithm.

In all decision situations, we typically have an incomplete learning curve, and want to extrapolate how the performance of the algorithm develops when more budget is provided.

Various model types are capable of extrapolating such curve segments, for example parametric models [9], meta-learning models [19] or specialized classifiers [16]. Parametric models are usually fit to the points of the curve segment and can then be used to make extrapolations. A popular parametric model used for learning curves is the inverse power law (IPL) [7, 13]. Another approach utilizes learning curves from earlier encountered datasets [19]. With this technique, learning curves constructed from other datasets, using the same algorithm, are combined to extrapolate the curve segment.

Mohr et al. [23] performed an in-depth comparison between parametric models for the extrapolation of learning curves. They empirically evaluate several parametric models over various datasets, and come to the conclusion that the IPL model is outperformed by the Morgan-Mercer Flodin model (MMF). This was an extension to the work of Gu et al. [9], who had previously performed this comparison on a smaller database with less parametric models and came to the opposite conclusion.

This paper complements the previous studies of Mohr et al. [23] by comparing parametric extrapolation with meta-learning based extrapolation. We provide insight to when it is beneficial to use a parametric model versus a meta-learning model across various extrapolation settings, which are defined by the available learning curve segment, the prediction target (the point to be extrapolated to) and the prediction objective. We do this by answering the following research questions:

- How does the curve segment (and thereby the availability of data) influence the performance of parametric and meta-learning models for the extrapolation of learning curves?
- How does the prediction target (i.e., the point towards which the learning curve needs to be extrapolated, this dictates the size of the learning curve) influence the performance of the aforementioned models?
- How is this influenced by the prediction objective (i.e., in some decision situations we need exact value prediction resulting in a regression task, and in others it suffices to pick the best learning algorithm

from a set of several options, resulting in a classification task.)

- Can this knowledge be applied to improve machine learning on a particle physics dataset?

To compare these two extrapolation model types we take the best performing parametric model according to current insights from the literature, namely MMF [9]. We will compare it to the Meta-Learning on Datasamples (MDS) model developed by Leite and Brazdil [19]. This is the only model we are aware of that utilizes learning curves from other datasets. As an additional baseline, we include a simple model that horizontally extrapolates from the last observation in the curve segment.

We find that using learning curves on other datasets with the meta-learning model is beneficial when only little parts of the learning curve have been observed. However, once more observations have been made, a parametric model, or just the last observation, outperforms the meta-learning model. Mohr et al. [23] found that MMF is the best performing parametric model in the setting that they considered, but in this work we show that by generalizing across settings, using a parametric model is in particular beneficial when the objective is to find the exact value of the learning curve itself. When the objective is to pick the best algorithm from a set of two algorithms, the surprising finding is that simply choosing the algorithm with the best score on the curve segment is better than picking the best one according to an extrapolation obtained from a parametric model, no matter how few observations have been made.

Chapter 2

Related Work

Learning curve extrapolation is performed in many different contexts of machine learning. Various methods have been developed in this regard. Our main focus is on learning curve extrapolation methods that can be used in general, these are usually tested on simple machine learning algorithms. In this section we show research done into learning curve extrapolation in the different contexts that we could find.

Firstly we cover the research done into modeling learning curves. Secondly, we will go over the meta-learning research done into extrapolating from other learning curves. Lastly, we will go over methods that use a probabilistic model to form uncertainties of learning curve extrapolations. These are usually used in hyperparameter optimization and neural architecture search.

2.1 Parametric models for learning curves

Learning curves have often been modelled by low parameter models. Power-law models have successfully been used in model selection to efficiently allocate resources to promising models [22]. They have also been used for early-stopping to stop the training of learning algorithms once it is highly probable that their performance will not significantly increase [13]. Theoretical works into the shapes of learning curves back up this line of work, as they suggest that learning curves usually have power-law behaviour, however, they also suggest that learning curves can have exponential behaviour [4, 12]. Mohr et al. [23] compare all the parametric models used for learning curves that they could find in the literature. They perform this comparison by looking at the model selection capabilities of each learning curve model. They find that, in practice, there are parametric models that can outperform power-law and exponential models.

Recently, research into learning curve models has received renewed attention in the domain of deep learning [11, 14]. It has been shown that power-law behaviour is also very common in this domain [10, 17].

2.2 Meta-learning models for learning curves

When completed learning curves are available, one can use these to extrapolate partial learning curves. Leite and Brazd [19] started this line of work with their Meta-Learning on Datasamples (MDS) method. To extrapolate a partial learning curve, they introduce a distance measure to datasets for which the completed learning curve is known. They use this to pick out the k closest learning curves. The mean of these k curves at the target is then taken as the extrapolation.

Leite and Brazd [20] build upon their work by also including meta-features of the datasets. These are features that describe the dataset and can be used to find similarities between datasets. . Rijn et al. [28] build on this work by introducing an algorithm that can rank a portfolio of classifiers.

Chandrashekaran and Lane [6] introduce a method that is very similar to MDS [19] for hyperparameter optimization. The main difference is that the distance measure is to completed learning curves on other hyerparameter settings instead of datasets.

2.3 Probabilistic models for learning curves

Since deep learning algorithms have many architecture and hyperparameter choices, finding the optimal choice is difficult. Hyperparameter optimization and neural architecture search are algorithmic approaches of solving this problem. Since the search space is usually continuous, or close to it, advanced methods apply a probabilistic model. This is done to make a confidence bound that dictates in what area of the search space to look next, thus narrowing down the search space. However, with the addition of learning curves these confidence bounds have also been used to discard sub optimal choices early in training. We cover some notable methods that use learning curves in this regard.

We note that, in this work, we do not include these probabilistic models, as our focus is on the extrapolation of learning curves outside of deep learning. In our case, the learning algorithms are also already assumed to be optimal. Due to this, the relationship between similar hyperparameters or architectures does not exist in our setting.

Synonymous to Leite and Brazdil [19], Chandrashekaran and Lane [6] pick completed learning curves that are similar to their partial learning curve, and make a probabilistic model from their extrapolation. The difference is that, in the context of hyperparameter tuning, they use a measure to find similar learning curves on already tested hyperparameter settings. Thus, these are on the same dataset.

Baker et al. [2] use support vector machines, random forests and simple linear regression to perform learning curve extrapolation for neural architecture search. They perform this regression not only on the partial learning curve, but also on meta features that dictate the architecture. This regression is then used to form confidence bounds.

A popular approach used in hyperparameter tuning and neural architecture search is Bayesian optimization [26]. This is used to form a surrogate model that predicts the performance of hyperparameters or network arhitectures. When combined with learning curves, this approach inherently uses previously completed learning curves to fit its surrogate model. Thus, this is a form of combining learning curve modelling and other learning curves. We will now cover methods that use Bayesian optimization with learning curves.

Domhan et al. [8] use a mixture of several learning curve models as their surrogate model. They use a prior that favors well behaved learning curves, which are saturating and increasing. This work has been extended by Klein et al. [16] who use a neural network to fit the weights of the mixture.

The work of Domhan et al. [8] has also been applied in neural architecture search by using learning curves of neural networks on different datasets [32].

Other surrogate models that have been used in the context of learning curves are gaussian processes [15, 27] and support vector machines [5].

Chapter 3

Background

Learning curves show the performance of a learning algorithm over dataset size. In this section, we will provide the mathematical background and definitions for them. These definition are adapted from Mohr and van Rijn [21]. Table 3.1 shows a summary of all the mathematical notation and definitions that we cover here.

At the end of this section we will apply this to LCDB to get the learning curves that we use.

3.1 Learning Curves

In supervised learning, a *task* $\mathcal{T}$ is formally given by the triplet $\mathcal{T} = (\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}, \mathcal{X}, \mathcal{Y})$, where $\mathcal{X}$ is the set of inputs called the *instance space*, $\mathcal{Y}$ is the set of output labels called the *label space*, and $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$ is a probability distribution over $\mathcal{X} \times \mathcal{Y}$. A *dataset* $d \subset \{(x,y) \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$ is made by sampling this probability distribution. Generally, it is assumed that this sampling is done identically and independently. We say that $\mathcal{D}$ is the set of all possible datasets. Using this, we can say that a *learning algorithm* is a function $a : \mathcal{D} \times \Omega \to H$, where Ω is a source of randomness, and $H = \{h \mid h : \mathcal{X} \to \mathcal{Y}\}$ is the set of all possible hypotheses.

For a learning curve we need the performance. Generally, the performance of a hypothesis is expressed using the (true) risk, or *out-of-sample error*:

$$\mathcal{R}_{out}(h) := \mathop{\mathbb{E}}_{(x,y)\sim\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}}[loss(y, h(x))] \tag{3.1}$$

We can extend this to learning algorithms. In the context of learning curves we are interested in the performance, given a specific size s for the dataset. By including this we get the following:

$$\mathcal{C}(a,s) := \mathop{\mathbb{E}}_{\omega\sim\mathbb{P}_\Omega, d\sim\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}, |d|=s}[\mathcal{R}_{out}(a(d,\omega))] \tag{3.2}$$

Where $\mathbb{P}_\Omega$ is the probability distribution of the random source Ω. The *learning curve* can now be defined as the function $\mathcal{C}(a,\cdot):\mathbb{N}\to\mathbb{R}$.

3.2 Empirical Learning Curves

In practice, it is not possible to calculate the out-of-sample error, as we usually do not have access to the probability distribution $\mathcal{P}_{\mathcal{X}\times\mathcal{Y}}$. Therefore, we need to use an estimator. We call this estimator the empirical risk, or *internal error*.

To calculate the internal error, we sample the distribution to get a dataset d. For a hypothesis h we then have:

$$\mathcal{R}_{in}(h)_d := \frac{1}{|d|}\sum_{(x,y)\in d} loss(y,h(x)) \tag{3.3}$$

Usually, we also use d for our learning algorithm to calculate h. Therefore, to forego some bias, it is beneficial to split the dataset into d_{tr} and d_{te}, such that $d_{tr}\cap d_{te}=\emptyset$, and $d_{tr}\cup d_{te}\subseteq d$. We use one for the learning algorithm and the other for the internal error. We say d_{tr} is the *train set*, used for the learning algorithm, while d_{te} is the *test set*, used for the internal error. From this, we define a *test error*:

$$\mathcal{R}_{in}(a(d_{tr},\omega))_{d_{te}} = \frac{1}{|d_{te}|}\sum_{(x,y)\in d_{te}} loss(y,a(d_{tr},\omega)(x)) \tag{3.4}$$

Similarly, we define a *train error*:

$$\mathcal{R}_{in}(a(d_{tr},\omega))_{d_{tr}} = \frac{1}{|d_{tr}|}\sum_{(x,y)\in d_{tr}} loss(y,a(d_{tr},\omega)(x)) \tag{3.5}$$

From equation 3.2, we see that the learning curve, at a given dataset size, is given by the expected value over the out-of-sample error. As we cannot calculate this directly, we need to use an estimator. The internal error already serves as an estimator for the out-of-sample error, so we can simply use the sample mean, as it is an unbiased estimator of the expected value for i.i.d. samples. This gives us the following estimator:

$$\hat{\mathcal{C}}(a,s)_D := \frac{1}{|D(s)|} \sum_{(d_{tr},d_{te},\omega)\in D(s)} \mathcal{R}_{in}(a(d_{tr},\omega))_{d_{te}} \tag{3.6}$$

Here we have a splitting technique $D : \mathbb{N} \to \mathcal{D} \times \mathcal{D} \times \Omega$, where $D(s)$ returns a subset of $\{(d_{tr}, d_{te}, \omega) \mid d_{tr} \cap d_{te} = \emptyset, d_{tr} \cup d_{te} \subseteq d, |d_{tr}| = s, \omega \in \Omega\}$. The idea of this splitting technique is that there are multiple methods of performing a split and keeping $|d_{tr}| = s$. One could split d in k folds, use a k'th as d_{te}, then use all combinations of size s of the rest for the different splits of d_{tr}. However, one could also change what fold is used for d_{te} in each split. If all combinations of folds for d_{te} and d_{tr} are used, and $d_{te} \cup d_{tr} = d$, then this is called a k-fold cross-validation.

Lastly, as is done by Mohr and van Rijn [21], the dataset size will be refered to as the *anchor point*. We will continue to denote it as s.

Table 3.1: *Notation and Definitions*

Notation	Meaning
d	**Dataset**, $d \subset \{(x,y) : x \in \mathcal{X}, y \in \mathcal{Y}\}$, where $\mathcal{X}$ is an instance space and $\mathcal{Y}$ a label space.
d_{tr}, d_{te}	**Train and test set**. One can define a test error for a learning algorithm a as $\mathcal{R}_{in}(a(d_{tr},\omega))_{d_{te}}$ and train error as $\mathcal{R}_{in}(a(d_{tr},\omega))_{d_{tr}}$. a always uses d_{tr}.
a	**Learning algorithm**, given by $a : \mathcal{D} \times \Omega \to H \texttt{ or } H^+$ ($\mathcal{D}$ set of all possible datasets, Ω source of randomness, H all possible hypotheses)
$\mathcal{R}_{out}$	**Out-of-sample error**, given by $\mathop{\mathbb{E}}_{(x,y)\sim\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}}[loss(y,h(x))]$
$\mathcal{C}(a,s)$	**Learning curve** for algorithm a at **anchor point** (dataset size) s, given by $\mathop{\mathbb{E}}_{\omega\sim\mathbb{P}_\Omega, d\sim\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}, \lvert d\rvert=s}[\mathcal{R}_{out}(a(d,\omega))]$
$\mathcal{R}_{in}(h)_d$	**Internal error** with respect to some dataset d. Given by $\frac{1}{\lvert d\rvert}\sum_{(x,y)\in d} loss(y,h(x))$
$\hat{\mathcal{C}}(a,s)_D$	**Empirical learning curve** for algorithm a at anchor point (dataset size) s given a set of random seeds Λ and a splitting technique D. Given by $\frac{1}{\lvert D(s)\rvert}\sum_{(d_{tr},d_{te},\omega)\in D(s)} \mathcal{R}_{in}(a(d_{tr},\omega))_{d_{te}}$.

3.3 Learning Curve Database (LCDB)

LCDB contains the performance of 20 unique learning algorithms and 248 unique datasets. However, not each learning algorithm was used on each dataset, thus, some datasets might contain less learning algorithms. In this section we will show how we use the data given by LCDB to form the learning curves.

For each dataset LCDB uses a splitting technique D[*] that creates 125 unique splits per anchor. Thus, for any given anchor s and dataset d in LCDB, the splitting technique D will create the sets given by $D(s) \subset \{(d_{tr}, d_{te}, \omega) \mid d_{tr} \cap d_{te} = \emptyset, d_{tr} \cup d_{te} \subseteq d, |d_{tr}| = s, \omega \in \Omega\}$ with $|D(s)| = 125$, for some source of randomness Ω. For each of these splits, LCDB shows the train and test error given by equation 3.5 and 3.4[†]. To get the value of the empirical learning curve $\hat{C}(a,s)_D$ we simply follow equation 3.6 and take the average over the 125 unique test errors. This is also referred to as a 5-fold Monte-Carlo Cross Validation (MCCV) [23].

This results in one unique empirical learning curve per dataset for each learning algorithm that was used on that dataset. However, we do not have the full empirical learning curve. LCDB contains only anchor points at sizes $\lceil 2^{\frac{7+k}{2}} \rceil$ for each $k \in \mathbb{N}$, until the maximum dataset size is reached. Additionally, there are anchor points representing certain percentages of the maximum dataset size.

[*]In addition to creating d_{tr} and d_{te} it also creates an extra set called the validation data. We can ignore it for our purpose.

[†]LCDB contains multiple different loss functions, we use the default which is accuracy.

Chapter 4

Extrapolation techniques

The extrapolation techniques that we use are: Morgan-Mercer Flodin (MMF), Meta-Learning on Datasamples (MDS) and the performance at the last available anchor. The last anchor is considered a baseline, which we refer to as 'Last'. In this section we will give an overview of these techniques. Figure 4.1 shows each technique used on a learning curve.

MMF is a parametric model that is used for the extrapolation of learning curves. It was picked over others as [23] found it to be the best performing parametric model in their analysis. MDS is an extrapolation technique that uses meta-data. This was picked as it uniquely does not use a parametric model, and has not been compared to techniques that use them. Last is a very simple extrapolation technique that is commonly used. We have included it as a baseline.

4.1 Morgan-Mercer Flodin (MMF)

MMF fits the parameters of the parametric model to the data in the curve segment. This parametric model is given by:

$$f_\theta = (ab + cx^d)/(b + x^d) \tag{4.1}$$

where $\theta := (a, b, c, d)$.

Given the anchor points $s_1, ..., s_m$ for a curve segment, the loss associated with parameters θ for the learning curve of some learning algorithm a is

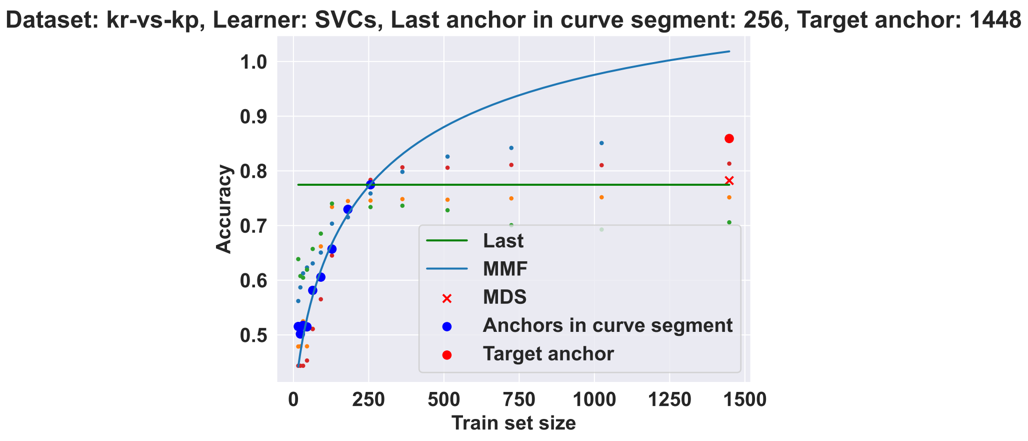


Figure 4.1: *Learning curve extrapolation performed by MMF, MDS and Last for the given curve segment and target anchor. The blue line is the extrapolation performed by MMF. The green line the extrapolation performed by Last. The smaller circles represent the k-nearest curves MDS uses to make its prediction, which is given by the red cross.*

defined as

$$loss(\theta) = \sum_{i=1}^{m} (\hat{C}(a, s_i) - f_\theta(s_i))^2 \cdot w_i, \tag{4.2}$$

where w_i is the weight given to anchor s_i. Here, we follow Mohr et al. [23] and set $w_i = 2^i$, which implies that the i-th observation is as important as *all* the observations prior to s_i together. Exploring other weighing schemes could be interesting future work.

Mohr et al. [23][*] use the Levenberg-Marquadt algorithm , we use the AdamW optimizer in PyTorch [24] to optimize for the parameters of the model instead. We found that this performs better.

4.2 Meta-Learning on Datasamples (MDS)

To predict the performance of a learner at a target anchor on dataset d, MDS [19][†] uses learning curves of this same learner from other datasets. This approach is parameter-free, so predictions can only be made for anchors for which the curve segments of the other datasets contain an observed performance. In particular, it is hence required that the other curves contain performances for the target anchor.

[*]The authors use the name mmf4 for the version of MMF that is used in our work
[†]The authors use the name A_MDS for the version of MDS used in our work

Among all the available curves of other datasets, the most relevant ones for the prediction task are determined based on the similarity of the already observed curve segments. Suppose that, for some learner a, the performances on dataset d have been observed for anchors $s_1, ..., s_m$. Then the (lack of) relevance of another dataset d' for the prediction task can be assessed through the dissimilarity of the learning curves, which in turn can be measured as the squared anchor-wise deviations:

$$R_a(d, d') = \sum_{i=1}^{m} (\hat{C}(a, s_i)_d - \hat{C}(a, s_i)_{d'})^2 \tag{4.3}$$

Given that the k nearest datasets are $d_1, ..., d_k$, MDS takes the mean performance of a at the target anchor s on these datasets and makes the extrapolation $\frac{1}{k} \sum_{i=1}^{k} \hat{C}(a, s)_{d_i}$.

Leite and Brazdil [19] improve their technique by scaling curves on other datasets before using the distance measure. Given a learning curve on a dataset $d' \neq d$, each point of that curve is multiplied by the following constant:

$$f = \frac{\sum_{i=1}^{N} (\hat{C}(a, s_i)_d \cdot \hat{C}(a, s_i)_{d'}) \cdot w_i)}{\sum_{i=1}^{N} (\hat{C}(a, s_i)_{d'} \cdot w_i)} \tag{4.4}$$

We also include this scaling in our implementation. The weighting mechanism used by the authors is given by $w_i = i^2$. However, we use the weighting mechanism Mohr et al. [23] uses as it improves performance and to apply the same weighing technique as used in MMF.

For the case of binary classification, one can refine the definition of dissimilarity since two learning curves are known. The technique applied here is to simply sum up the algorithm-wise dissimilarities:

$$R_{a_1,a_2}(d, d') = \sum_{i=1}^{m} (\hat{C}(a_1, s_i)_d - \hat{C}(a_1, s_i)_{d'})^2 + \sum_{i=1}^{m} (\hat{C}(a_2, s_i)_d - \hat{C}(a_2, s_i)_{d'})^2 \tag{4.5}$$

Two remarks are due with respect to the computation of nearest neighbors. First, we use a value of $k = 4$ in our experiments as it provides the best results in preliminary experiments. However, it is conceivable that neighbors are not chosen based on a fixed number but that they must meet a certain quality (or at least not be significantly worse than the nearest neighbor). Second, the curve scaling in Eq. 4.4 could be applied before or after the computation of nearest neighbors. We apply it before the computation of the neighbors.

This work has been extended in several directions, i.e., to determine the best learning algorithm from multiple options and to include meta-features [20], as well as to include different scoring criteria including run time [28].

4.3 The last anchor baseline

A trivial baseline is to extrapolate the known part of the learning curve simply with a horizontal line at the performance of the last available anchor. Accordingly, we call this baseline "Last" as was done by Mohr et al. [23]. For any target anchor, Last predicts the value of the largest anchor point in the given curve segment. Similar to the other two techniques, this prediction can either be used in a binary or regression prediction objective.

Chapter 5

Experiments

We will describe the experimental setup, results and discuss these[*].

5.1 Experimental Setup

The performance of an extrapolation technique depends on the context of the extrapolation setting. The context is defined by (at least) the following variables:

- **Target anchor.** The anchor point to extrapolate to. Does not have to be a specific anchor point if, for example, the limit performance is of interest.
- **Curve segment.** The anchor points for which the curve is already given. Extrapolation is performed from this segment to the target anchor.
- **Prediction objective.** What the extrapolation is used for. The extrapolation can be an intermediate step in a comparison between various learning algorithms. In this case, the only result of interest is what learning algorithm will perform better at the target anchor, essentially making this a classification task. We will explore binary versions of this task i.e., where the best algorithm out of a pair of two must be selected. In other cases, we are interested in the exact performance value, essentially making this a regression task.

[*]Full details: `https://github.com/quantagenmtg/AMLthesis`

- **Metadata.** The type of additional data that can be used for the extrapolation task. For example, a meta-dataset of completed learning curves for the learning algorithms of interest on other datasets.

Our analysis is based on the Learning Curve Database (LCDB) [23]. It contains learning curves of 20 learning algorithms on 248 unique datasets. The extent to which we analyse each context variable is detailed below.

5.1.1 Target anchor

To get a balanced overview of how the extrapolation techniques work in different extrapolation settings, we consider a broad set of target anchors. For each of 248 datasets and each of the 20 algorithms, LCDB contains scores for anchor points at sizes $\lceil 2^{\frac{7+k}{2}} \rceil$ with $k \in \{1, 2, ..\}$ until the maximum dataset size is reached. We pick the anchor points that are present in at least half the datasets (i.e., leaving out anchor points at higher values that are only obtained at the larger datasets) as possible target anchors. This leaves us with the following anchor points:

$$\{16, 23, 32, 45, 64, 91, 128, 181, 256, 362, 512, 724, 1024, 1448, 2048, 2896, 4096\}$$

We do not consider the first anchor point (16) as a target anchor, as we need at least one anchor point in the curve segment to make a prediction. The remaining anchors will be considered candidates for the target anchor.

5.1.2 Curve segment

Intuitively, we assume that the extrapolation techniques work better when presented with performance results at more anchors and performance results at larger anchors. To investigate the exact dynamics, we will explore various curve segments including a range of anchors.

We start off with a curve segment containing just the anchor point 16. We add 23 to get a curve segment that includes 16 and 23. We continue adding the next anchor point until we have 16 different curve segments, each having one additional anchor over the previous one. We don't include the anchor point 4096 as then there would be no possible target anchor.

5.1.3 Prediction objective

There are mainly two possible prediction objectives in the context of learning curves. The first is to use the known curve segment of a learner to predict its performance at some target anchor. This is a regression task and is important for data-acquisition to predict how much the performance can improve with more data. The second objective is to use the curve segments of *two* algorithms to predict which of them will exhibit the better performance at the target anchor. This is a binary classification task and important for early discarding in model selection.

However, instead of addressing the classification problem through binary classification, we compare the extrapolated and determine the one with the best extrapolated performance at the target anchor.

We will analyse how each extrapolation technique performs in the binary and regression prediction objectives. We have limited ourselves to these prediction objectives as these can be addressed with the introduced extrapolation techniques.

5.1.4 Metadata

The metadata that we are interested in is other learning curves. Specifically, we would like to know if using learning curves from other datasets is better than using a parametric model. To this extent we compare MDS to MMF. Note that we specifically presume that we do not have learning curves from the same dataset, which is required for the learning curve Bayesian neural networks [16].

techniques that use other metadata, such as dataset meta-features and features describing the learning algorithms, also exist. For example, Leite and Brazdil [20] have extended MDS to include some other relevant metadata. We do not include these in our analysis as we are purely interested in the use of other learning curves.

5.2 Metrics

Let s be some target anchor and a some learning algorithm (or a_1, a_2 two learning algorithms that are being compared). If the estimated and true prediction performances of a at anchor s are given by $\hat{y}(a,s)$ and $\hat{C}(a,s)$ respectively, then the loss of the extrapolation model is defined as:

- Absolute error $(a,s) = |\hat{y}(a,s) - \hat{C}(a,s)|$
- Binary error $(a_1,a_2,s) = \begin{cases} 0 & \text{if } \arg\max_i \hat{y}(a_i,s) = \arg\max_i \hat{C}(a_i,s) \\ 1 & \text{else} \end{cases}$

- Risk $(a_1, a_2, s) = \begin{cases} 0 & \text{if } \arg\max_i \hat{y}(a_i, s) = \arg\max_i \hat{C}(a_i, s) \\ |\hat{C}(a_1, s) - \hat{C}(a_2, s)| & \text{else} \end{cases}$

The absolute error is used in the regression prediction objective while the binary error and risk are used in the binary prediction objective. We use these two metrics in the binary objective as we are interested in (1) how often the extrapolation techniques pick the worse learning algorithm and (2) the loss in performance if that learning algorithm were used instead of the better one.

Chapter 6

Results and Discussion

We show our results in a series of figures. Each incrementing figure has an additional aggregation. Additional figures can be found in Appendix A.

Our main results are found in the last figure, Figure 6.3. However, a lot of underlying information is lost in this figure because of aggregations. We thus start by showing the results for specific extrapolation setting and then generalizing step by step. For each figure we either show the performance and / or the relative performance of the extrapolation techniques. The relative performance takes two extrapolation techniques then subtracts their performance from each other.

The 20 algorithms from LCDB on which the analysis is based on are shown in table 6.1

6.1 Relative performances of extrapolation techniques for a fixed curve segment and target anchor

Figure 6.1 shows the relative performance of the three extrapolation techniques for the binary prediction objective for a fixed target anchor size of 4096 and a fixed curve segment. The size of the largest anchor in this curve segment is 724. This is shown individually for each possible pair of learning algorithms, we refer to these pairs as the "binary pairings". Additionally, the relative performances are also summarized in boxplot distributions over the binary pairings. For each comparison "A vs B" the

Table 6.1: *Learning algorithms and their abbreviations*

learning algorithm	abreviation
SVClinear	SVCl
SVCpoly	SVCp
SVCrbf	SVCr
SVCsigmoid	SVCs
ExtraTreesClassifier	xTrs
GradientBoostingClassifier	GrBo
RandomForestClassifier	rFor
LogisticRegression	LogR
PassiveAggressiveClassifier	PaAg
Perceptron	Perc
RidgeClassifier	Ridg
SGDClassifier	SGD
BernoulliNB	Bern
MultinomialNB	MuNo
KNeighborsClassifier	KNei
MLPClassifier	MLP
DecisionTreeClassifier	dTre
ExtraTreeClassifier	xTre
LinearDiscriminantAnalysis	linD
QuadraticDiscriminantAnalysis	QuaD

binary error of B is subtracted from the binary error of A at each binary pairing. After this the mean is taken over the datasets. Positive values (red) favor B while negative values (blue) favor A. The sets of these means are then used to make the boxplots.

We see from the boxplots that, for this particular curve segment and target anchor, Last clearly outperforms MDS and MMF. This can be seen as the means and median are well above the 0 line.

We also see that in the comparison between MMF and MDS there is a slight advantage for MMF as the mean and median are slightly above the 0 line.

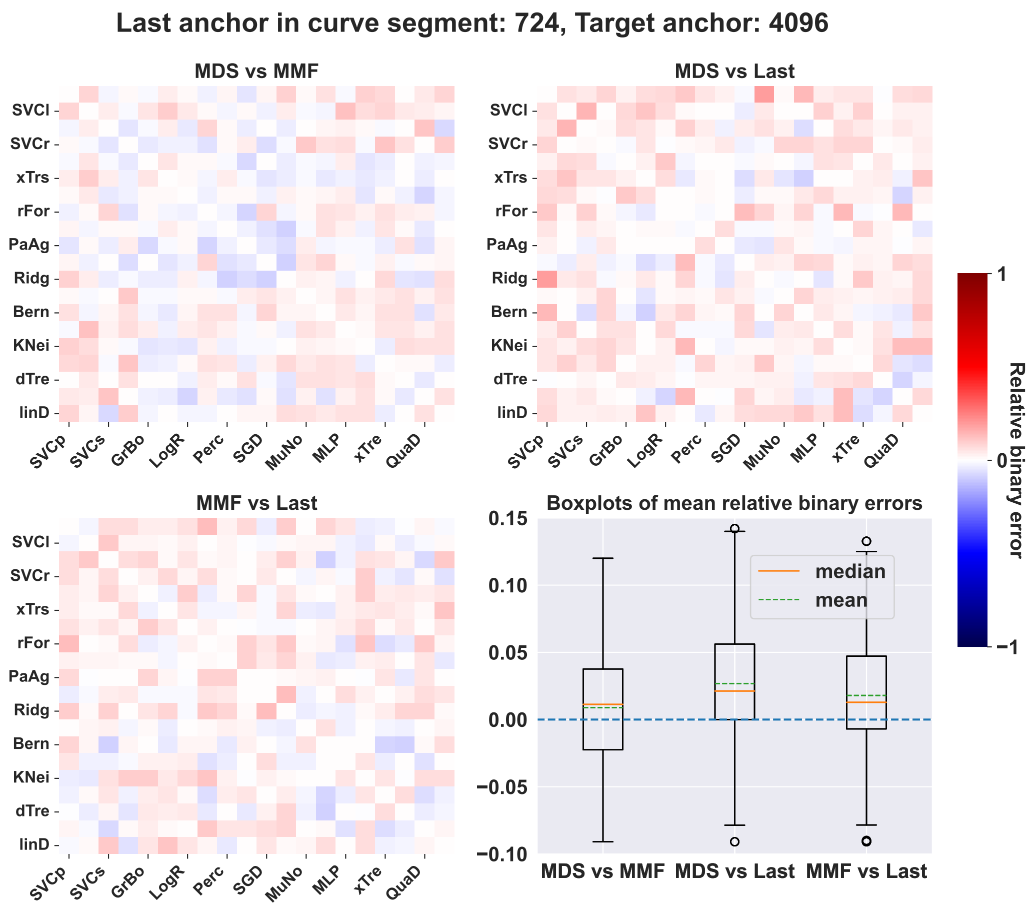


Figure 6.1: *Relative performances of the extrapolation techniques for a fixed curve segment with largest anchor of size 724, and fixed target anchor of size 4096. Shown are the results for the binary prediction objective with binary error as metric. The colorbar refers to the figures in the left column and the top right figure. Names of learning algorithms are abbreviated.*

6.2 Performances of extrapolation techniques for increasing curve segments and a fixed target anchor

Figure 6.2 now offers a slightly more generalized view on the situation than Figure 6.1 by varying over different curve segments. The target anchor remains fixed at a size of 4096. The left and middle columns are for the binary prediction objective, and the right column for the regression prediction objective. The left column uses the risk as metric, the middle

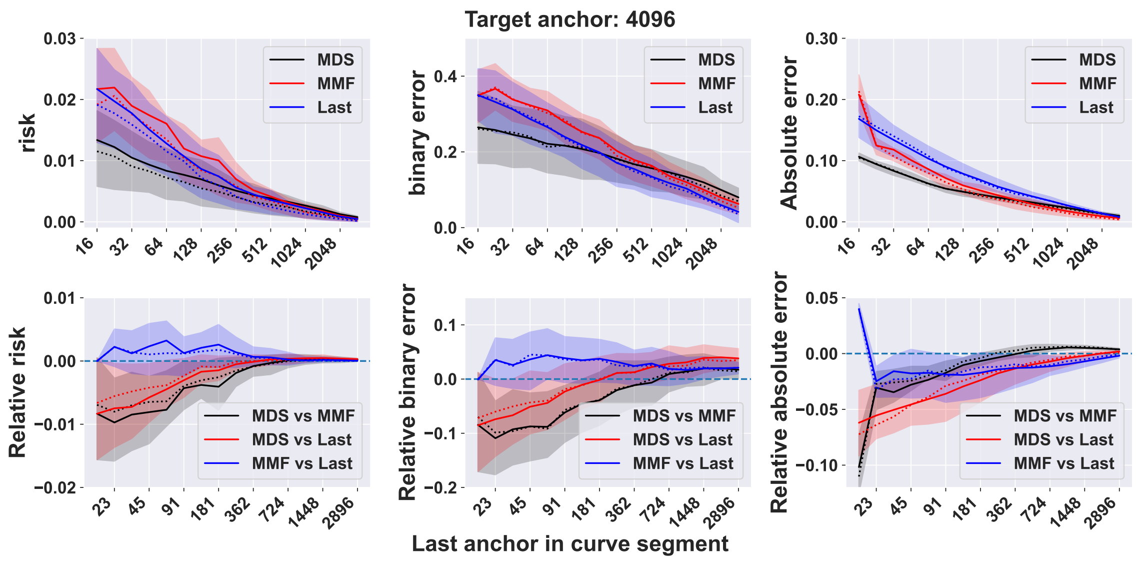


Figure 6.2: *Performances of extrapolation techniques for moving curve segments and a fixed target anchor of size 4096. Thick lines represent the average, dotted lines the median, and the shaded areas show the interquartile performances.*

the binary error, the right the absolute error. Results are aggregated by first computing the mean over the datasets per binary pairings (as in Figure 6.1) or per learners (for the regression prediction objective). The means obtained for these binary pairings or learners are then aggregated again in the plot as follows; thick lines represent the average, dotted lines the median, and the shaded areas show the interquartile performances. The top row are the individual performances of the extrapolation techniques, and the bottom row are relative performances between each technique. In the bottom row, a comparison of "A vs B" means that positive values favor B while negative values favor A. For these results the metric values of B are subtracted from A before any aggregation is performed.

We ignore the very first value in the regression prediction objective for MMF and any comparison with MMF, this is because with only one anchor in the curve segment MMF is unable to perform an extrapolation.

We see that MMF outperforms Last on average for the regression prediction objective but is outperformed by Last on the binary prediction objective. For regression, the quartiles suggests Last does outperform MMF at times at some smaller curve segments. The reason for this is because there are quite a few learning curves in LCDB that descend at the beginning and then start rising at a later anchor point. When only the descending part of the learning curve is available MMF will extrapolate into the wrong direction. In these cases another parametric model should be used. Even so, at smaller curve segments an improvement of around $0 - 4\%$ accuracy can be expected for the prediction.

For the binary prediction objective we see that MMF never improves over Last even at smaller curve segments. By looking at the quartiles we see that for any curve segment Last outperforms MMF for around 75% of cases. Last has quite a high error at lower curve segments, larger than 30%. However, it seems that MMF cannot improve on this.

Furthermore we see that MDS seems to outperform both MMF and Last on smaller curve segments. For the regression prediction objective we can also see that MDS is a lot more precise and accurate at these smaller curve segments. It starts at an error of around 0.1 and has closer quartiles than the other techniques. As the learning curves represent accuracy, this means that MDS can predict the accuracy of a learning algorithm within $\pm 10\%$ with one anchor point. For the binary prediction objective we see that the binary error of MDS is still quite high at the lower curve segments, but it stays about 5-10% lower than the other techniques. Even though the binary error is large here we see that the risk is under 1.5%, which is quite low. This shows that even though MDS predicts wrong over 20% of times at lower curve segments, when it predicts wrong the loss in accuracy is only under 1.5%.

6.3 Relative performances of extrapolation techniques for increasing curve segments and target anchors

Figure 6.3 now further generalizes the previous view by additionally ranging over different target anchor sizes. The leftmost column again shows the risk for each binary pairing per extrapolation technique. Similarly to before, a comparison of "A vs B" means that the risk of B is subtracted from the risk of A. The mean of this is then taken over the datasets and the binary pairings. Positive values (red) favor B while negative values (blue) favor A. For the second column the same aggregation is performed but for the binary error. For the last column the same is done for the absolute error, and instead of the binary pairings the mean is taken over the learners as there are no binary pairings in the regression prediction objective. The colorbar shows two different scales, the left side is used for the risk values

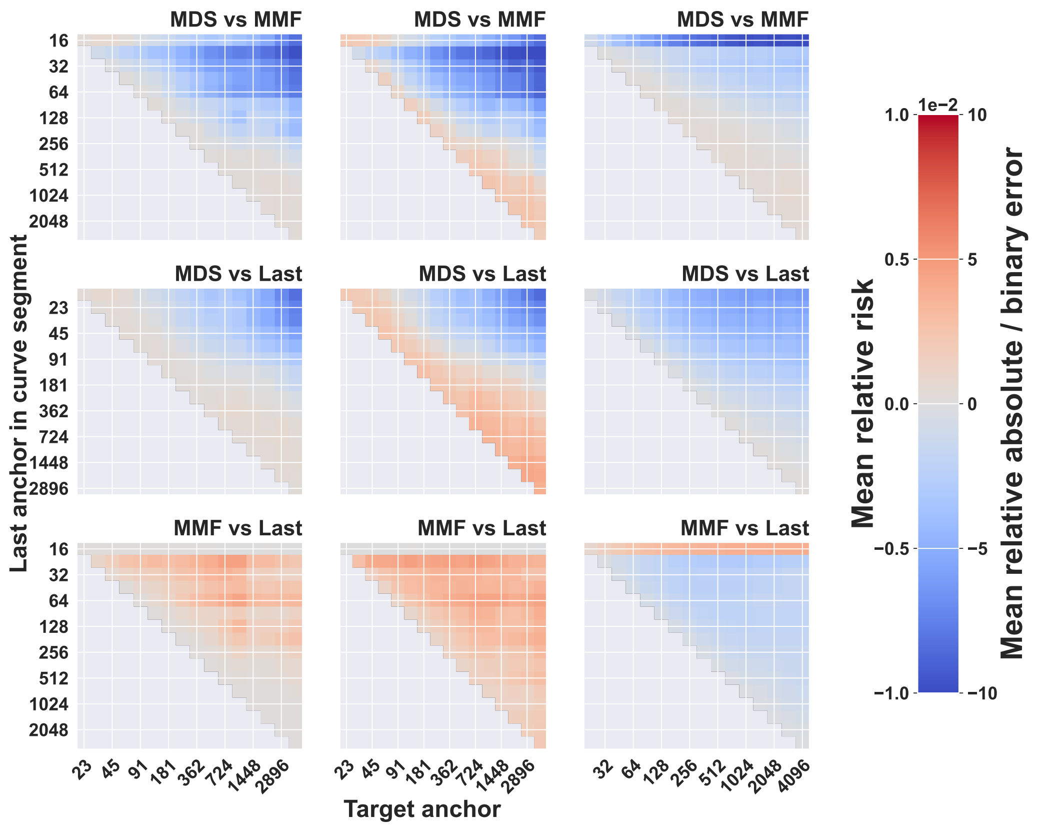


Figure 6.3: *Relative performances of extrapolation techniques for moving curve segments and target anchors. The left column uses the risk as metric, the middle the binary error, the right the absolute error. The left side of the colorbar is for the risk and the right side for both binary and absolute error.*

and the right for the absolute and binary error values. Values falling outside of the bounds given by the colorbar have been set to the value of the nearest bound. As in the previous section, for the regression prediction objective we ignore the curve segment that only contains the anchor 16 for any comparison that include MMF.

From this figure we see that the trend identified in the previous section occurs for most target anchors. MDS outperforms both MMF and Last on smaller curve segments. However, the smaller the target anchor the smaller the anchor point at which Last or MMF become better than MDS. This is because making the target anchor smaller means less observations are needed to complete the learning curve, thus smaller curve segments give more information than they would for larger target anchors. This shows that a meta-learning model that uses other learning curves is useful when there is little information about the learning curve. As a rough estimate we see that when the last anchor in the curve segment is under a 5th or 10th the size of the target anchor, MDS performs better than the other two extrapolation techniques.

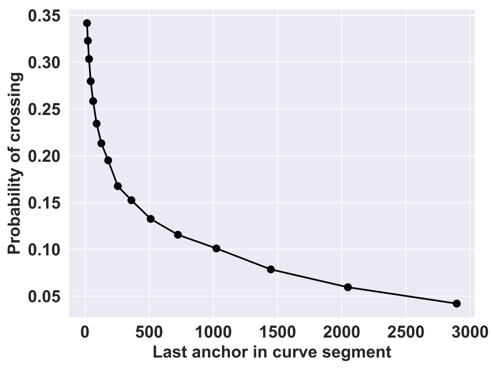
It is clear that it is not beneficial to use a parametric model for the binary prediction objective. This is because MMF is outperformed by simply using only the last anchor in the curve segment. We see that as more data on the learning curve becomes available the last anchor becomes a better estimation of the target anchor and thus outperforms MDS.

For the regression prediction objective it is better to use a parametric model over just the last anchor. At larger curve segments MMF will slightly outperform MDS. Thus, as more data on the learning curve becomes available it is better to use a parametric model if a high accuracy of prediction is required.

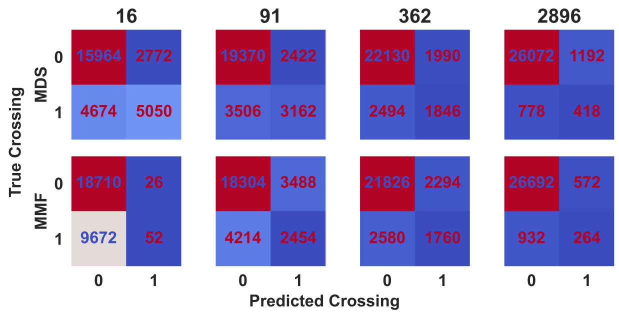
6.4 Learning curve crossing

To explain why the parametric model is worse than using Last in the binary prediction objective, we analyse learning curve crossing. We say that two learning curves cross if the better performing algorithm is different at the largest target anchor 4096 and largest anchor point in the curve segment. Last always predicts that learning curves will not cross due to its horizontal extrapolation. Using the error of Last, we can check how probable learning curve crossing is. This can be seen in Figure 6.4a. This figure shows the probability that two learning curves will cross beyond the last anchor in the curve segment. We only check if the learning curves have crossed between the last anchor point in the curve segment and the anchor point 4096, our largest target. We can also check how well the other two techniques predict learning curve crossing. This can be seen in Figure 6.4b. This figure shows confusion matrices at the largest target for different curve segments. The largest anchor in the curve segment is stated above the respective matrices. 1 means learning curves cross after the curve segment, 0 means they don't. The matrices are normalized over the rows.

Looking at Figure 6.4a we see that, in general, learning curves are unlikely to cross. For the smallest curve segment there is already less than a 35% chance. It becomes more unlikely that learning curves will cross the larger the curve segment. This relationship seems to form an exponential decay.



(a) Probability of Learning Curve Crossing



		16 (0)	16 (1)	91 (0)	91 (1)	362 (0)	362 (1)	2896 (0)	2896 (1)
MDS	0	15964	2772	19370	2422	22130	1990	26072	1192
MDS	1	4674	5050	3506	3162	2494	1846	778	418
MMF	0	18710	26	18304	3488	21826	2294	26692	572
MMF	1	9672	52	4214	2454	2580	1760	932	264

(b) Confusion Matrices

Figure 6.4: *(a) Shown are the number of learning curves in LCDB that cross after the given curve segment, normalized. (b) The confusion matrices show how how well MDS and MMF predict learning curve crossing per curve segment. The values shown are the number of curves in LCDB that fall under that quadrant.*

Figure 6.4b gives us some insight why MMf cannot improve over Last. When there is no learning curve crossing MMF will usually detect this, over 90% of times for any curve segment. When there is learning curve crossing MMF will usually not detect this, under 40% of times for most curve segments. Since there are a lot more learning curves that do not cross Last is better, as it always gets those correct. The times when learning curves do cross and MMF detects this do not make up for this loss, as MMF often does not detect the crossing of learning curves.

Figure 6.4b also shows us why MDS is better at smaller curve segments. We can see that at smaller curve segments MDS correctly predicts learning curve crossings around 45-50% of times, which is better than MMF. It also detects when learning curves do not cross around 85-90% of times similarly to MMF. Since learning curves are more likely to cross at smaller windows than larger windows it is much more useful to detect this for smaller windows. The 10-15% loss over Last for falsely predicting learning curve crossing is thus compensated by the fact that learning curve crossing happens more at smaller curve segments, and MDS can correctly predict this half of the time.

We thus see that a meta-dataset of other learning curves is useful to detect learning curve crossing when there are little observations available, and less is known about the learning curve. However, as more data on the learning curve becomes available learning curve crossing is much less likely to occur. This means predicting that learning curves will never cross is better than risking the false negatives of MDS or MMF.

Chapter 7

HIGGS dataset

We apply the learning curve extrapolation techniques on the HIGGS dataset [31]. When particles collide at high energies in a collider there is a possibility of rare particles to appear [3]. To find when this happens one has to distinguish between noise and signal, to this extent machine learning is used. HIGGS is a Monte Carlo simulated dataset providing measurements of a collider and functions of these measurements. Each instance is classified as either noise or signal.

To check if our results in the previous chapter apply to the HIGGS dataset we make learning curves with the same 20 learning algorithms that are used in LCDB. The anchor points will be the same as the ones from Section 5.1.1.

We thus perform the same analysis as before but on 1 dataset instead of 248. For MDS we will be using the remainder of the datasets from LCDB as the meta-dataset.

Figure 7.1 shows the relative performances of the extrapolation techniques across each curve segment and target anchor similar to Figure 6.3 but only for the HIGGS dataset. The leftmost column shows the risk. As before, a comparison of "A vs B" means that the risk of B is subtracted from the risk of A. The mean of this is then taken over the the binary pairings. Positive values (red) favor B while negative values (blue) favor A. For the second column the same aggregation is performed but for the binary error. For the last column the same is done for the absolute error, and instead of the binary pairings the mean is taken over the learners as there are no binary pairings in the regression prediction objective. The colorbar shows two different scales, the left side is used for the risk values and the right for the absolute and binary error values. Values falling outside of the bounds given by the colorbar have been set to the value of the nearest bound. As in the previous chapter, for the regression prediction objective we ignore the curve segment that only contains the anchor 16 for any comparison that include MMF.

There are more anchor points in this figure as we have added all the anchors present in LCDB for the HIGGS dataset. We also note that for the larger target anchors MDS has very little curves to pick from, as not a lot of curves in LCDB contain those anchor points. Specifically for the last target anchor there are no other curves in LCDB that also contain that anchor point, thus no data is shown for this anchor in any comparison that includes MDS.

This figure is not as clear as Figure 6.3 as we only regard one dataset. However, we can still clearly see that the results from the previous chapter also apply to the HIGGS dataset, which are as follows:

- MDS outperforms both MMF and Last on smaller curve segments
- Either Last or MMF outperform MDS on larger curve segments
- Last outperforms MMF in the binary prediction objective on any curve segment except the first

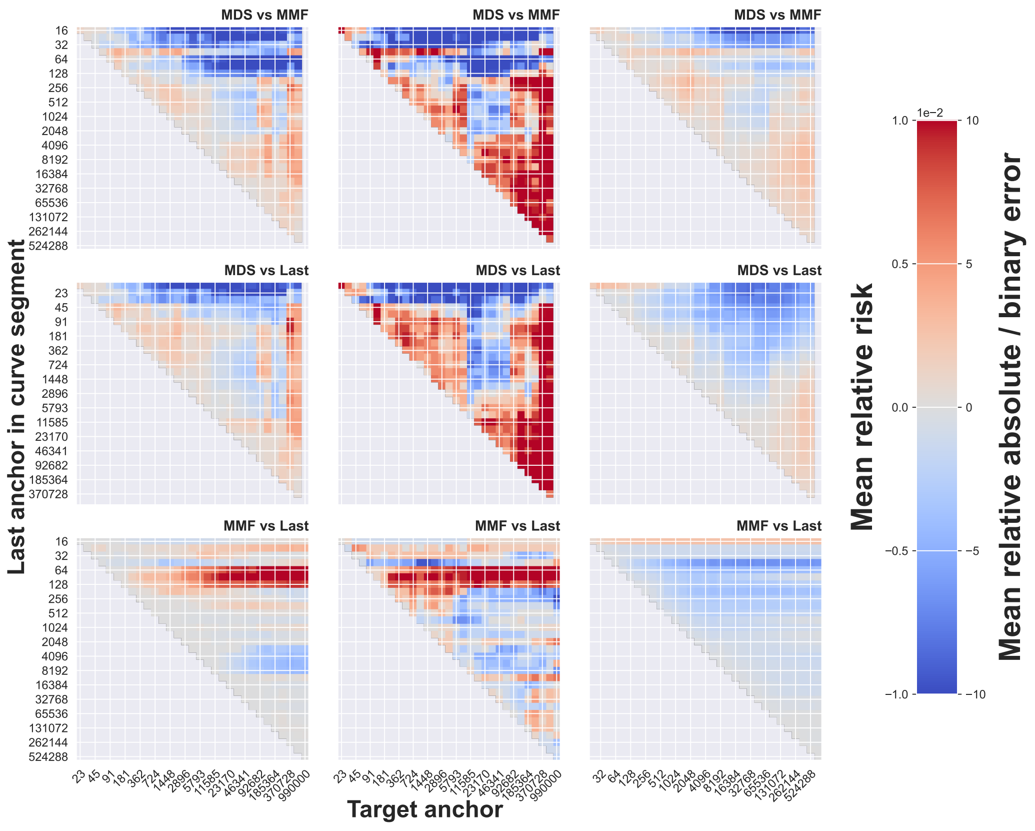


Figure 7.1: *Relative performances of extrapolation techniques for moving curve segments and target anchors solely on the HIGGS dataset. The left column uses the risk as metric, the middle the binary error, the right the absolute error. The left side of the colorbar is for the risk and the right side for both binary and absolute error.*

Chapter 8

Conclusion

In this paper we have compared two distinct learning curve extrapolation techniques, i.e., the parametric model MMF and the meta-learning model MDS. Additionally we have included a baseline which takes the performance at the last anchor of the learning curve as its prediction.

We have performed this comparison across different extrapolation settings. In these settings we vary the availability of data for the extrapolation, the size of the target anchor, and the prediction objective.

From this comparison we find that, for both prediction objectives, MDS is the better performing extrapolation technique when little information is available on the learning curve. While due to the amount of settings it is hard to exactly quantify, loosely speaking MDS will perform better than the other techniques when the largest anchor in the curve segment is up to a 5th or 10th the size of the target anchor. Once more data becomes available, both MMF and the baseline generally outperform MDS. This goes to show that, with more data available it is better to rely on a parametric model, or just the last anchor, and with less data available it is better to rely on a meta-learning model.

We find that, when predicting which of two learning algorithms is better, the parametric model is often outperformed by just using the information available on the last anchor. However, this is not the case when the objective is to find the exact value of the learning curve at the target. In that case the parametric model outperforms using just the last anchor in almost all cases.

Lastly we find that these results also hold true for the HIGGS dataset and can thus be used in the context of machine learning for particle physics.

Interesting further research could be

1. A conditional analysis that depends on the learners.
2. Including other parametric models in the analysis.
3. To study the impact of weights in the techniques. Here we used an exponential decay of older anchors.
4. Performing the evaluation not in terms of curve segment / target anchor combinations, but in therms of difficulty.
5. Including multi-class case in the analysis, where the goal is to predict a ranking of the learning curves.

Bibliography

[1] Kim Albertsson, Piero Altoe, Dustin Anderson, Michael Andrews, Juan Pedro Araque Espinosa, Adam Aurisano, Laurent Basara, Adrian Bevan, Wahid Bhimji, Daniele Bonacorsi, Paolo Calafiura, Mario Campanelli, Louis Capps, Federico Carminati, Stefano Carrazza, Taylor Childers, Elias Coniavitis, Kyle Cranmer, Claire David, Douglas Davis, Javier Duarte, Martin Erdmann, Jonas Eschle, Amir Farbin, Matthew Feickert, Nuno Filipe Castro, Conor Fitzpatrick, Michele Floris, Alessandra Forti, Jordi Garra-Tico, Jochen Gemmler, Maria Girone, Paul Glaysher, Sergei Gleyzer, Vladimir Gligorov, Tobias Golling, Jonas Graw, Lindsey Gray, Dick Greenwood, Thomas Hacker, John Harvey, Benedikt Hegner, Lukas Heinrich, Ben Hooberman, Johannes Junggeburth, Michael Kagan, Meghan Kane, Konstantin Kanishchev, PrzemysÅaw KarpiÅski, Zahari Kassabov, Gaurav Kaul, Dorian Kcira, Thomas Keck, Alexei Klimentov, Jim Kowalkowski, Luke Kreczko, Alexander Kurepin, Rob Kutschke, Valentin Kuznetsov, Nicolas KÃ¶hler, Igor Lakomov, Kevin Lannon, Mario Lassnig, Antonio Limosani, Gilles Louppe, Aashrita Mangu, Pere Mato, Helge Meinhard, Dario Menasce, Lorenzo Moneta, Seth Moortgat, Meenakshi Narain, Mark Neubauer, Harvey Newman, Hans Pabst, Michela Paganini, Manfred Paulini, Gabriel Perdue, Uzziel Perez, Attilio Picazio, Jim Pivarski, Harrison Prosper, Fernanda Psihas, Alexander Radovic, Ryan Reece, Aurelius Rinkevicius, Eduardo Rodrigues, Jamal Rorie, David Rousseau, Aaron Sauers, Steven Schramm, Ariel Schwartzman, Horst Severini, Paul Seyfert, Filip Siroky, Konstantin Skazytkin, Mike Sokoloff, Graeme Stewart, Bob Stienen, Ian Stockdale, Giles Strong, Savannah Thais, Karen Tomko, Eli Upfal, Emanuele Usai, Andrey Ustyuzhanin, Martin Vala, Sofia Vallecorsa, Justin Vasel, Mauro Verzetti, Xavier VilasÃs-

Cardona, Jean-Roch Vlimant, Ilija Vukotic, Sean-Jiun Wang, Gordon Watts, Michael Williams, Wenjing Wu, Stefan Wunsch, and Omar Zapata. Machine learning in high energy physics community white paper. *Journal of Physics: Conference Series*, 1085(2):022008, sep 2018.

[2] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating neural architecture search using performance prediction. In *6th International Conference on Learning Representations, ICLR'18*, 2018.

[3] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5(1):4308, 2014.

[4] Olivier Bousquet, Steve Hanneke, Shay Moran, Ramon van Handel, and Amir Yehudayoff. A theory of universal learning. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 532–541. ACM, 2021.

[5] Andrés F. Cardona-Escobar, Andrés Felipe Giraldo-Forero, Andrés Eduardo Castro-Ospina, and Jorge Alberto Jaramillo-Garzón. Efficient hyperparameter optimization in convolutional neural networks by learning curves prediction. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, volume 10657 of *Lecture Notes in Computer Science*, pages 143–151. Springer, 2017.

[6] Akshay Chandrashekaran and Ian R. Lane. Speeding up hyperparameter optimization by extrapolation of learning curves using previous builds. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017*, volume 10534 of *Lecture Notes in Computer Science*, pages 477–492. Springer, 2017.

[7] Corinna Cortes, Lawrence D. Jackel, Sara A. Solla, Vladimir Vapnik, and John S. Denker. Learning curves: Asymptotic values and rate of convergence. In *Advances in Neural Information Processing Systems 6*, pages 327–334. Morgan Kaufmann, 1993.

[8] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 3460–3468. AAAI Press, 2015.

[9] Baohua Gu, Feifang Hu, and Huan Liu. Modelling classification performance for large data sets. In *Advances in Web-Age Information Management, Second International Conference, WAIM 2001*, volume 2118 of *Lecture Notes in Computer Science*, pages 317–328. Springer, 2001.

[10] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *CoRR*, abs/1712.00409, 2017.

[11] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022.

[12] Marcus Hutter. Learning curve theory. *CoRR*, abs/2102.04074, 2021.

[13] George H. John and Pat Langley. Static versus dynamic sampling for data mining. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 367–370. AAAI Press, 1996.

[14] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[15] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, volume 54 of *Proceedings of Machine Learning Research*, pages 528–536. PMLR, 2017.

[16] Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. In *5th International Conference on Learning Representations, ICLR'17*, 2017.

[17] Tobit Klug and Reinhard Heckel. Scaling laws for deep learning based image reconstruction. In *The Eleventh International Conference*

on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023.

[18] Mark Last. Improving data mining utility with projective sampling. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 487–496. ACM, 2009.

[19] Rui Leite and Pavel Brazdil. Predicting relative performance of classifiers from samples. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005)*, volume 119 of *ACM International Conference Proceeding Series*, pages 497–503. ACM, 2005.

[20] Rui Leite and Pavel Brazdil. Active testing strategy to predict the best classification algorithm via sampling and metalearning. In *ECAI 2010 - 19th European Conference on Artificial Intelligence*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 309–314. IOS Press, 2010.

[21] Felix Mohr and Jan N. van Rijn. Learning curves for decision making in supervised machine learning – a survey. *CoRR*, abs/2201.12150, 2022.

[22] Felix Mohr and Jan N. van Rijn. Fast and informative model selection using learning curve cross-validation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(8):9669–9680, 2023.

[23] Felix Mohr, Tom J. Viering, Marco Loog, and Jan N. van Rijn. LCDB 1.0: An extensive learning curves database for classification tasks. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2022*, volume 13717 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2022.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[25] Foster J. Provost, David D. Jensen, and Tim Oates. Efficient progressive sampling. In *Proceedings of the Fifth ACM SIGKDD Interna-*

tional Conference on Knowledge Discovery and Data Mining, pages 23–32. ACM, 1999.

[26] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[27] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *CoRR*, abs/1406.3896, 2014.

[28] Jan N. van Rijn, Salisu Mamman Abdulrahman, Pavel Brazdil, and Joaquin Vanschoren. Fast algorithm selection using learning curves. In *Advances in Intelligent Data Analysis XIV*, volume 9385 of *Lecture Notes in Computer Science*, pages 298–309. Springer, 2015.

[29] Tom Viering and Marco Loog. The shape of learning curves: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7799–7819, 2022.

[30] Gary M. Weiss and Ye Tian. Maximizing classifier utility when there are data acquisition and modeling costs. *Data Mining and Knowledge Discovery*, 17(2):253–282, 2008.

[31] Daniel Whiteson. HIGGS. UCI Machine Learning Repository, 2014.

[32] Martin Wistuba and Tejaswini Pedapati. Inductive transfer for neural architecture optimization. *CoRR*, abs/1903.03536, 2019.

Appendix A

Additional figures

In this appendix we provide additional and alternative results. For the regression setting we provide plots for the first and second aggregation individually for each learner. We also provide an alternative to Figure 6.2 where the mean is taken over the learners binary pairs instead of the datasets. Lastly we provide the absolute performance instead of the relative performances seen in Figure 6.3.

In Figures A.1 to A.6 we use boxplots to show the distribution of the absolute error for 3 different fixed curve segments and a fixed target anchor size of 4096. This distribution is taken over the 248 datasets.

In Figures A.7 to A.10 we again fix the target anchor to a size of 4096 but now we increase the curve segments. Instead of the entire distribution we only show the mean over the datasets. This results in a curve which we show individually for each learner.

In Figure A.11 we take the mean over the learners in the regression prediction setting, and binary pairings in the binary prediction setting. The distribution is then shown over the datasets. This is done the other way round in Figure 6.2.

In Figure A.12 we display the absolute performances of the extrapolation techniques instead of the relative performances, as was done in Figure 6.3. As before, blue indicates a low value and red a large value.

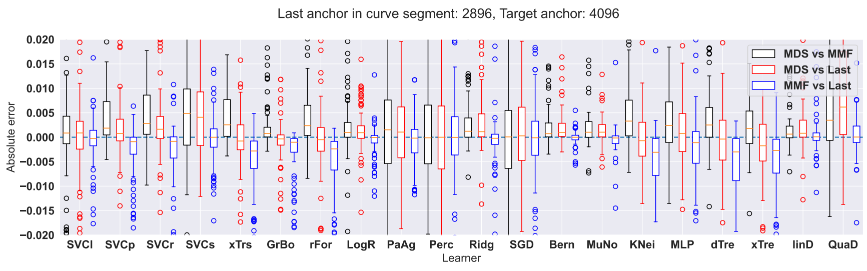


Figure A.1: *Relative performances of extrapolation techniques per learner for fixed target and curve segment in the regression prediction objective.*

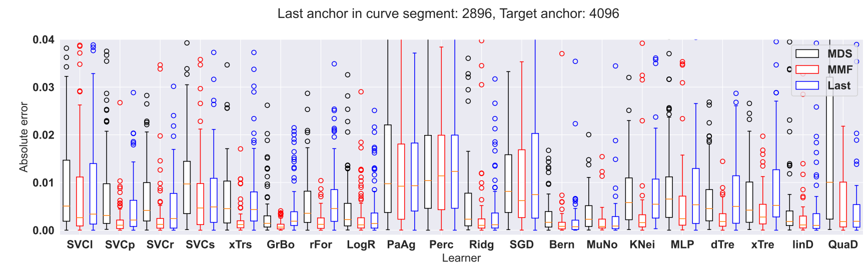


Figure A.2: *Absolute performances of extrapolation techniques per learner for fixed target and curve segment in the regression prediction objective.*

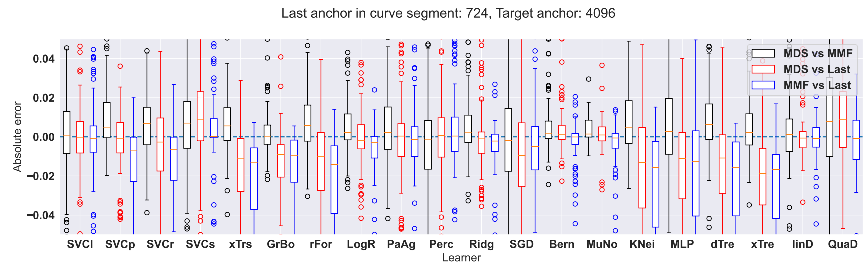


Figure A.3: *Relative performances of extrapolation techniques per learner for fixed target and curve segment in the regression prediction objective.*

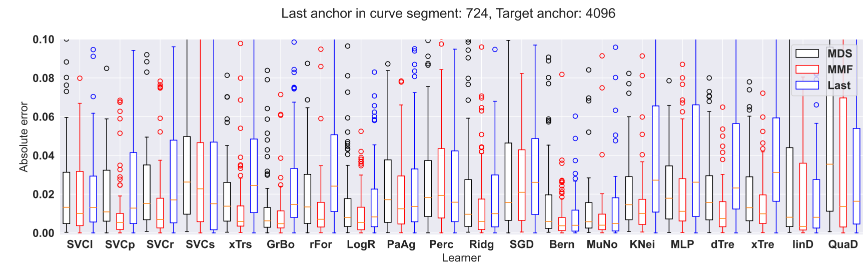


Figure A.4: *Absolute performances of extrapolation techniques per learner for fixed target and curve segment in the regression prediction objective.*

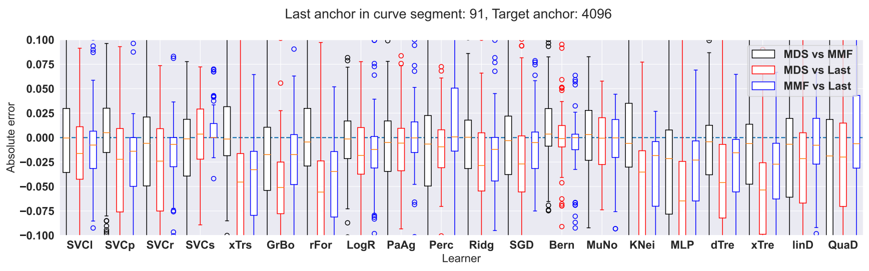


Figure A.5: *Relative performances of extrapolation techniques per learner for fixed target and curve segment in the regression prediction objective.*

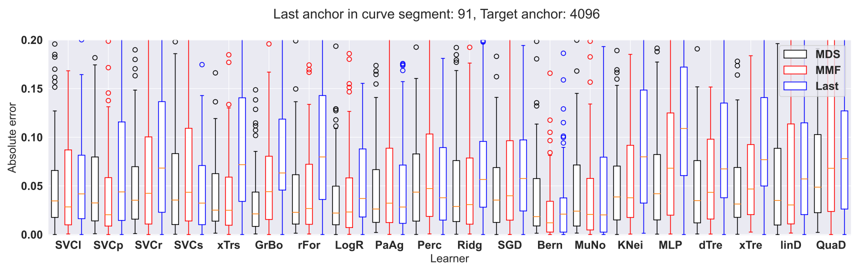


Figure A.6: *Absolute performances of extrapolation techniques per learner for fixed target and curve segment in the regression prediction objective.*

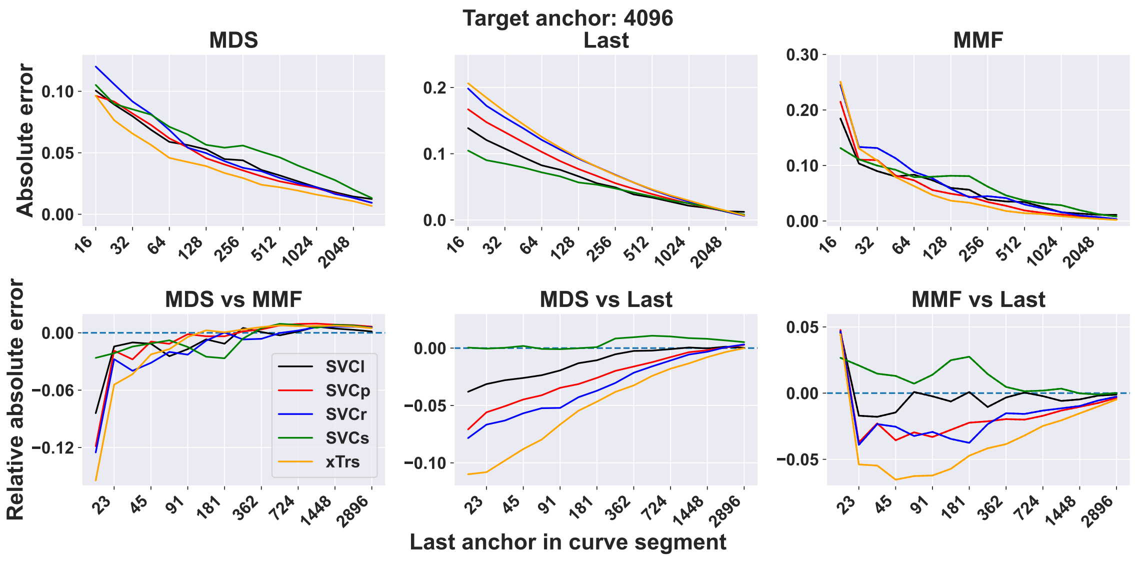


Figure A.7: *Individual performances of extrapolation techniques for moving curve segments and a fixed target anchor in the regression prediction objective for 5 learners.*

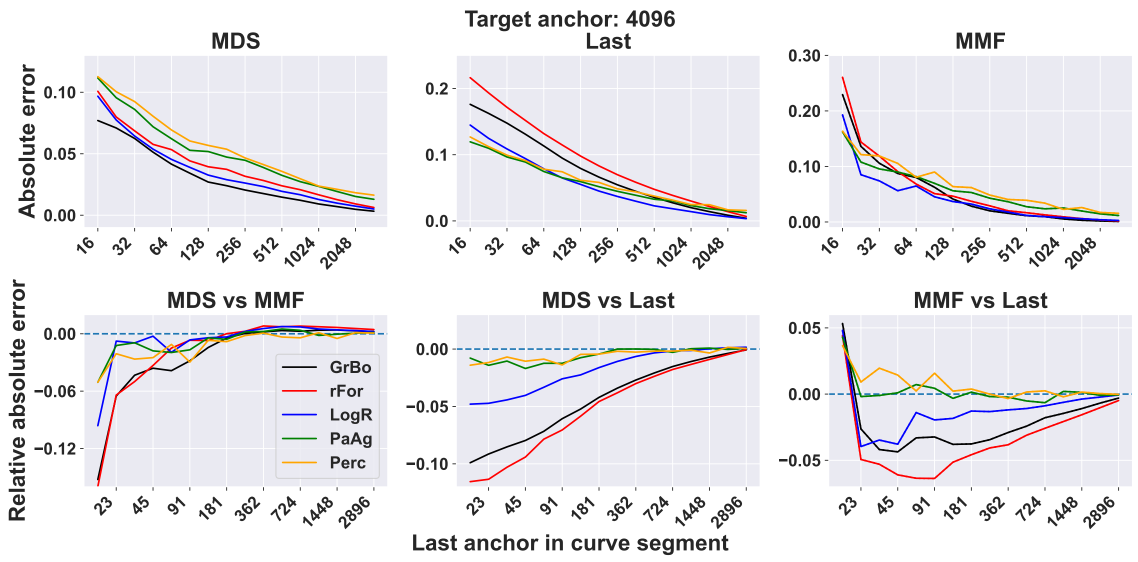


Figure A.8: *Individual performances of extrapolation techniques for moving curve segments and a fixed target anchor in the regression prediction objective for 5 learners.*

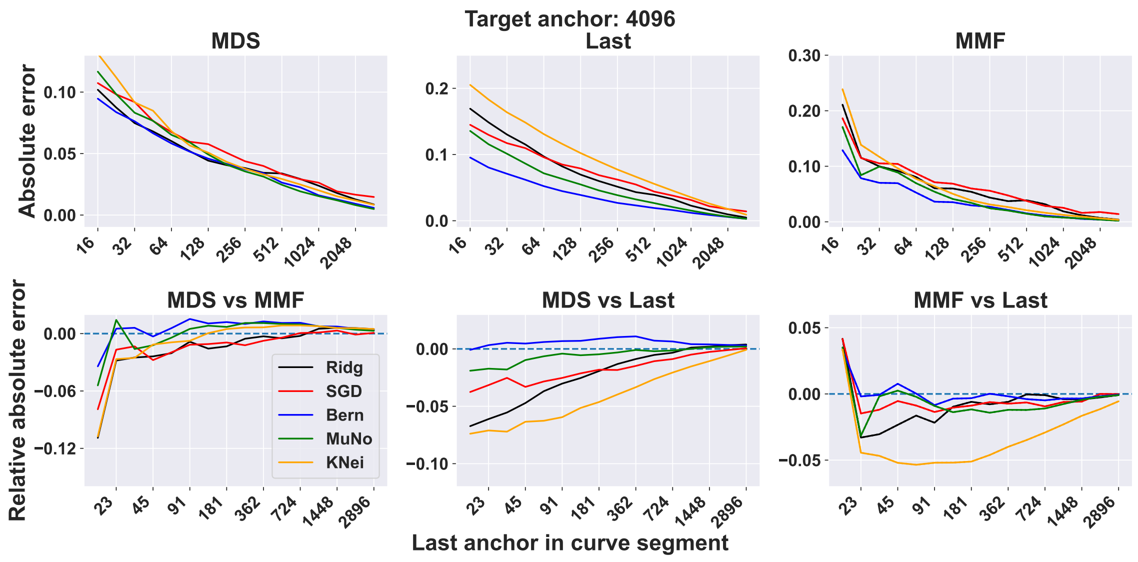


Figure A.9: *Individual performances of extrapolation techniques for moving curve segments and a fixed target anchor in the regression prediction objective for 5 learners.*

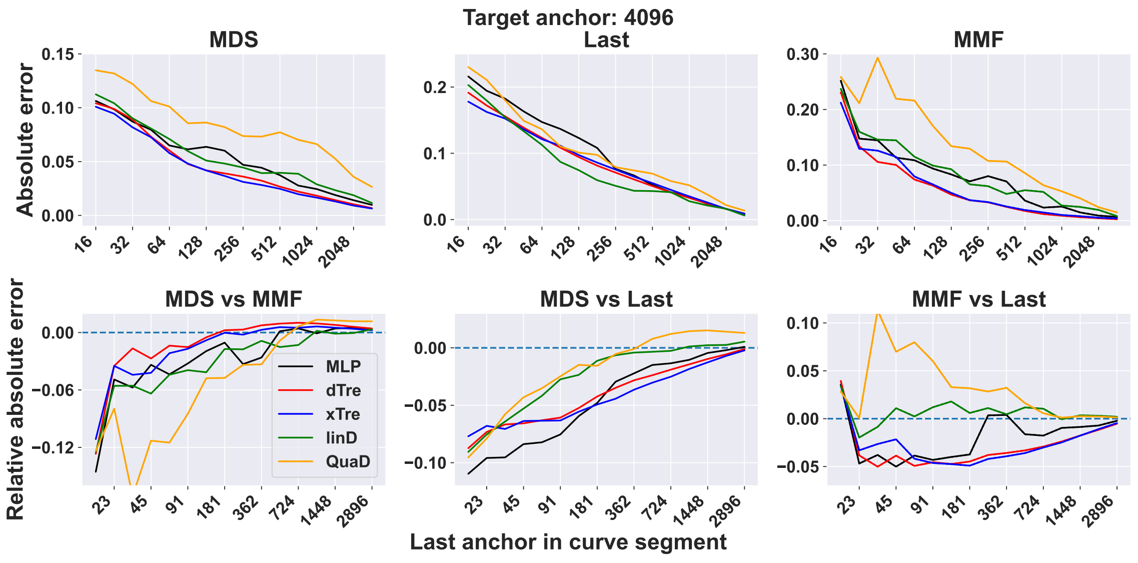


Figure A.10: Individual performances of extrapolation techniques for moving curve segments and a fixed target anchor in the regression prediction objective for 5 learners.

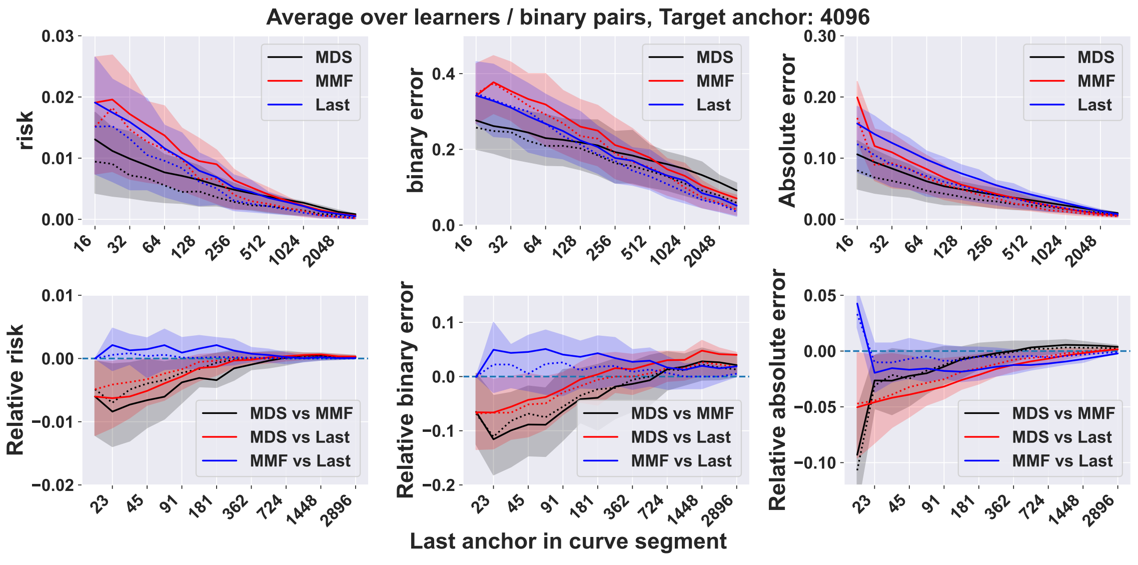


Figure A.11: Performances of extrapolation techniques for moving curve segments and a fixed target anchor of size 4096. Thick lines represent the average, dotted lines the median, and the shaded areas show the interquartile performances.

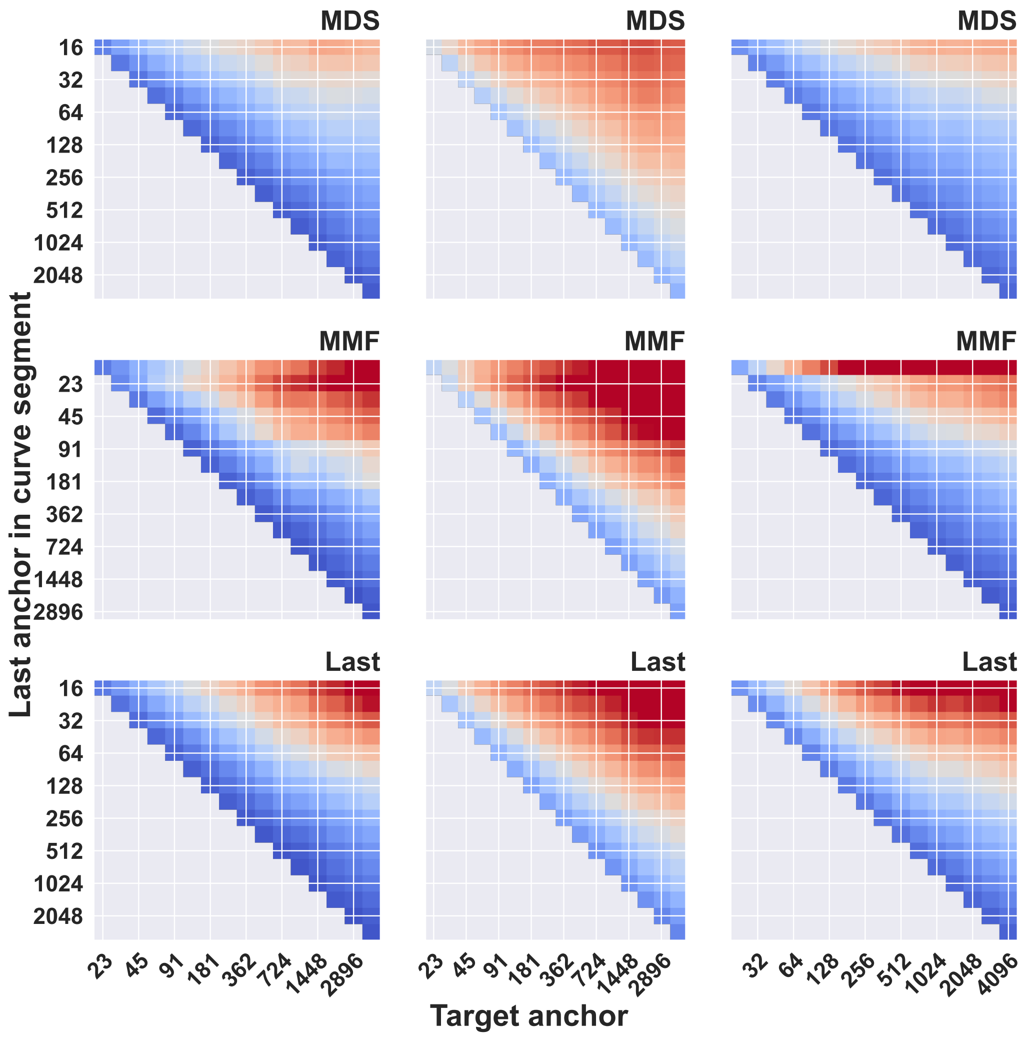


Figure A.12: *Absolute performances of extrapolation techniques for moving curve segments and target anchors. The left column uses the risk as metric with an upper bound of* 0.02*, the middle the binary error with an upper bound of* 0.3*, the right the absolute error with an upper bound of* 0.15*. Each has a lower bound of* 0*.*