Opleiding Informatica

Building an experimental learning curve database from many configurations of the gradient boosting algorithm

Rivaldo das Neves Brito

Supervisors:
J.N. van Rijn & F. Mohr

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

27/12/2025

Abstract

Learning curves show the relationship between an algorithm's performance and the amount of training data, shedding light on its learning behaviour. This information proves valuable in machine learning tasks such as model selection or data acquisition. To use learning curves for these purposes, it is crucial to study and improve our understanding of the wide range of learning curve behaviours and shapes. Consequently, previous studies contributed to this goal by producing learning curve databases, enabling researchers to study learning curves more effectively. Our thesis contributes by building a learning curve database that uniquely focuses on learning curves for various hyperparameter configurations of the gradient boosting algorithm. We describe the process of creating the database and briefly analyse the learning curves to study their behaviour. To showcase the value of our learning curve database, we conduct two experiments focused on machine learning tasks. In these experiments, we observed that the gradient boosting surrogate outperformed both the random forest and ridge models in predicting learning curves for unseen configurations. Separately, we found that mmf4 performed better than random forest and the baseline in terms of accuracy, while random forest outperformed the others in ranking performance. Although we succeeded in producing a learning curve database and demonstrated its potential to support research, future work could improve various aspects of the database. The experiments also have room for refinement. Future research would improve the setup, include statistical testing, and focus more on the behaviour and shapes of curves.

Contents

1 Introduction

In machine learning, the learning curve is a widely used tool for visualising an algorithm's learning behaviour. It typically depicts the performance relative to a specified resource. We denote two types of learning curves [10]. Iteration-wise curves illustrate a model's performance over the number of training iterations (e.g., epochs), as commonly seen with neural networks. Sample-wise curves instead depict the performance for different amounts of training data.

Recent studies highlight the utility of learning curves in machine learning tasks [10, 16]. For instance, in model selection, learning curves allow us to estimate which models will perform better than others. This enables us to discard less promising candidates early and allocate resources to those with greater potential. Furthermore, methods such as learning curve extrapolation can help determine the value of training models with additional resources, which allows us to estimate the minimum resources needed to obtain a sufficiently performing model.

Learning curves do not always show the same shapes and behaviours, which means using them for these tasks is not as straightforward. For instance, it becomes easier to extrapolate learning curves when they align well with a parametric formula. However, extrapolating models may struggle when the curves exhibit ill behaviour. Various types of ill-behaviours exist, as displayed in Figure 1, and studying these different learning-curve behaviours is essential if we want to utilise learning curves effectively in machine learning.

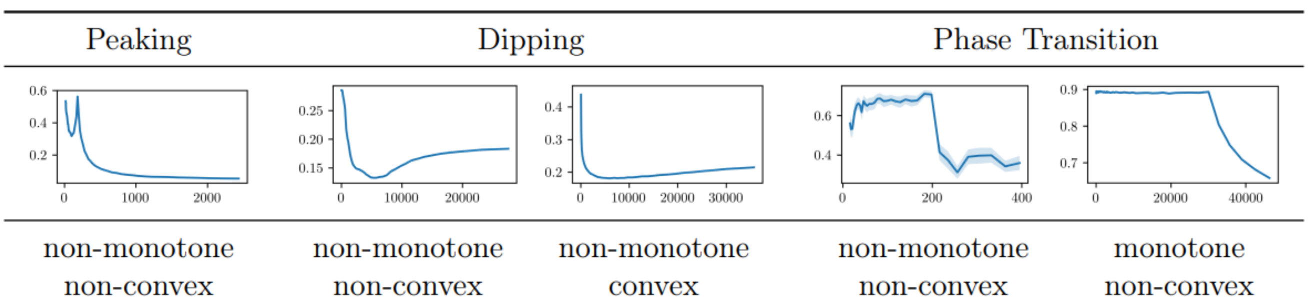


Figure 1: Ill-behaved learning curves on OpenML CC-18 datasets, plotting validation error against training set size. They illustrate phenomena such as peaking, dipping, and sudden phase transitions. Taken from [17].

Several empirical learning curve databases were developed to facilitate studies on learning curves. The most extensive of these is the Learning Curve Database (LCDB) [11], containing sample-wise curves for a variety of algorithms and datasets. They built on the original database in LCDB 1.1 [17] , improving the methodology and addressing issues from the previous version.

However, existing databases tend to focus on many different algorithms and datasets at the cost of using only the default or a few distinct hyperparameter configurations per algorithm. With our thesis, we want to build a learning curve database that does the opposite.

1.1 Contributions

The primary purpose of our thesis is to develop a learning curve database for various configurations of the gradient boosting algorithm and demonstrate its utility in machine learning tasks such as model selection and data acquisition.[1]

We focus on a single algorithm, gradient boosting [4], and produce 1,000 unique configurations for each of ten classification datasets. Thus, we limit the number of algorithms and datasets to create a large quantity of configurations. For each configuration, we perform nested cross-validation and record the validation and test scores for different amounts of training data. This allows us to build sample-wise learning curves for each dataset, where each curve corresponds to a unique configuration.

In addition to a brief visual analysis of our learning curves, we conduct two experiments where our learning curves are used for a specific machine learning task to validate the utility of our database.

In the first experiment, we train surrogate models on our learning curves and their corresponding configurations. Then we analyse and compare how the trained models predict learning curves for unseen configurations. This task addresses hyperparameter optimisation; how surrogate models that map configurations to learning curves can be leveraged to predict the performance of other configurations.

For the second experiment, we extrapolate partial curves of various sizes to a fixed target, using different types of models. We evaluate and compare these models using several performance metrics, enabling us to study their strengths and weaknesses for this task.

1.2 Thesis outline

In Section 2, we review related work on learning curves and their use in supervised learning, including works on learning curve databases. Section 3 describes the database design and implementation. Additionally, it contains a brief visual analysis of the created learning curves. Section 4 presents the experiments, including the setups, results, and analyses. Section 5 concludes with a summary of the thesis, listing the key findings and results of our work. It also discusses the limitations and possible improvements for future research.

[1] Database available: https://github.com/Rivdnb/Experimental-learning-curve-database

2 Related Work

In this section, we introduce the use of learning curves in machine learning, specifically supervised machine learning, and discuss relevant work that has inspired this thesis. We also explain how our thesis contributes compared to other similar research.

2.1 Literature reviews: Use cases & Curve characteristics

Two recent literature reviews provide complementary perspectives on learning curves in supervised learning. They extensively discuss potential applications of learning curves in research and conclude that further research is needed to deepen our understanding.

The authors of [10] provide a survey oriented towards methods that use or extrapolate learning curves to support decision-making in supervised learning. They categorise literature along three axes: the decision-making situation, the technical question at hand, and the resources used for modelling the learning curve.

In contrast, the authors of [16] comprehensively discuss the shapes and properties of learning curves, such as monotonicity, convexity, and peaking. They dive into the origins of learning curves and related terminology, and reinforce the importance of understanding these shapes for research and other applications.

Illustrated in Figure 2, they list at least three different decision-making situations in which learning curves serve as a crucial instrument.

1. Quantitative data acquisition: This situation entails determining the required amount of data to achieve a satisfactory performance. We can extrapolate learning curves to estimate the performance benefit of acquiring additional training data.

2. Early stopping of model training: It can be valuable to stop training as soon as possible to minimise the use of training resources. For instance, if we can predict when a model's performance becomes satisfactory, we can stop training early and save on resources.

3. Early discarding in model selection: Learning curve extrapolation enables us to estimate which models may perform better than others down the line. We can apply this in model selection, where we aim to discard non-competitive candidates as early as possible. Successive halving [6] and Hyperband [8] are examples of techniques that exploit learning curve information by evaluating and discarding candidates at different points of the curves.

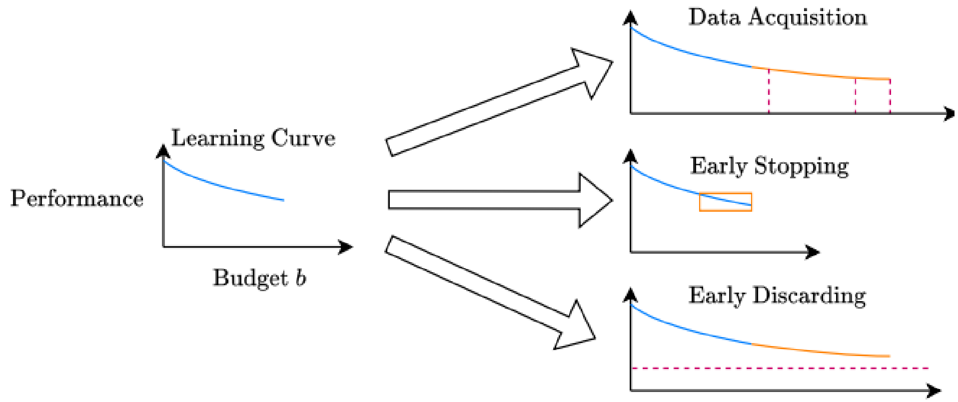


Figure 2: According to [10], there are three types of decision-making situations in which learning curves are commonly used. In these figures, the x-axis represents the budget in a given unit, and the y-axis represents performance measured by a given metric. Taken from [10].

They also separate modelling approaches utilising learning curve models from those that do not. Learning curve models fit functional relationships to a partial curve to extrapolate the curve. Curve properties, such as smoothness and monotonicity, can affect the performance of these models. To cope with this, practical implementations contain counter-measures such as restarts, parameter bounds, and discarding poor fits.

However, other approaches can also answer many questions regarding learning curves. Surrogate models [3] serve as a relevant example for our thesis. These models can use information from other learning curves to predict learning curve values for different learners.

2.2 Learning curves databases

The main inspiration for this thesis is the Learning Curves Database (LCDB). This research aimed to enable robust studies of learning curves to increase our shared understanding of their behaviour and shapes. Two different iterations of the LCDB were published:

In LCDB 1.0 [11], the authors created an initial database of sample-wise learning curves generated from various classification algorithms and datasets. They used the database to analyse the presence of learning curve behaviours. They concluded that the majority of learning curves do not exhibit ill-behaviour but instead display behaviours such as monotonicity and convexity.

However, the authors of the next iteration, LCDB 1.1 [17], found these conclusions premature and performed a deeper analysis on the frequency of various learning curve behaviours. Through this analysis, they aimed to discover how these behaviours affect tasks such as curve fitting and model selection. They also improved the database by addressing previous limitations. Some examples of improvements include: adding feature scaling, handling missing data, and improving the learning curve resolution.

The authors found a significant amount of ill-behaviour using their improved LCDB 1.1, which contrasts with the findings in the LCDB 1.0 paper. They described their findings on curve ill-behaviours, discussing their frequency and differences for various learning algorithms. They conclude that many causes of ill-behaviour remain unknown, and hope that their database enables researchers to investigate them.

This thesis draws substantial inspiration from the LCDB papers, particularly in their methods for creating learning curves. However, the LCDB papers and our thesis differ in multiple aspects. Aside from the scope of the study, the main difference lies in that our database contains curves from many configurations of a single learning algorithm. In contrast, their database consists of curves for many different learning algorithms but only deploys their default hyperparameters.

Another difference lies in their experiments, which focus on studying the behaviour and characteristics of learning curves. Instead, our experiments focus on showcasing the utility of our learning curve database in supervised learning tasks.

Other notable works on learning curve database include the LCBench [18], Task-Set [9], and BUTTER [2]. They focus on epoch-wise curves for neural networks, making them incomparable to this thesis and the LCDB papers, which instead focus on sample-wise curves.

3 Development of the learning curve database

This section covers the methods and details of creating the learning curve database. We focused on only one learning algorithm: gradient boosting. This algorithm incrementally combines multiple base learners to form a prediction. We chose the scikit-learn [12] GradientBoostingClassifier implementation, which uses regression trees as base learners.

This implementation has multiple hyperparameters that influence the algorithm's functioning and performance. We measure performance for different hyperparameter values while also varying the amount of training data. As a result, we created learning curves for different hyperparameter values.

3.1 Attributes of a learning curve

Each learning curve has a corresponding hyperparameter configuration. We describe a hyperparameter configuration in our case as a 6-tuple of hyperparameter values $(lr, est, md, mss, msl, ss)$. In other words, a combination of values assigned to these six hyperparameters is called a hyperparameter configuration, or a *configuration* for short. Table 1 shows the description and possible values of the hyperparameters. The ranges partially align with recent work [13].

Hyperparameter	Description	Data type	Range	Sampling
Learning_rate (lr)	Decides how much an individual tree contributes to the predictions.	float64	[0.01,0.2]	log-uniform
N_estimators (est)	The number of base learners used for boosting.	int64	{50,51,...,300}	uniform
Max_depth (md)	The maximum tree depth of the base learners.	int64	{3,4,...,20}	uniform
Min_samples_split (mss)	The minimum samples required to split an internal node.	int64	{2,3,...,8}	uniform
Min_samples_leaf (msl)	The minimum samples required at a leaf node.	int64	{1,2,3,4}	uniform
Subsample (ss)	The fraction of samples used to fit the base learners.	float64	[0.5,1.0]	uniform

Table 1: The gradient boosting hyperparameter selection. We give the value range, data type, sampling method, and a short description for each hyperparameter. Hyperparameters as taken from scikit-learn [12].

We produced the configurations by randomly sampling values within the ranges shown in Table 1. Our learning curves show the cross-validation performance of a configuration across different training data sizes. We specifically implemented nested cross-validation, as described in more detail in Section 3.4.

We refer to the different training set sizes as *anchors*, which aligns with recent work [11]. We formally define the anchors as follows.

$$n_k \rightarrow \{ \lceil 16 \cdot 2^{k/2} \rceil \mid k \in \mathbb{Z}^+ \} = [16, 23, 32, ...]$$

We generated the learning curves for our database by averaging the cross-validation accuracies for each anchor. Figure 3 shows an example of a produced learning curve with the mean cross-validation accuracy and standard deviation given per anchor.

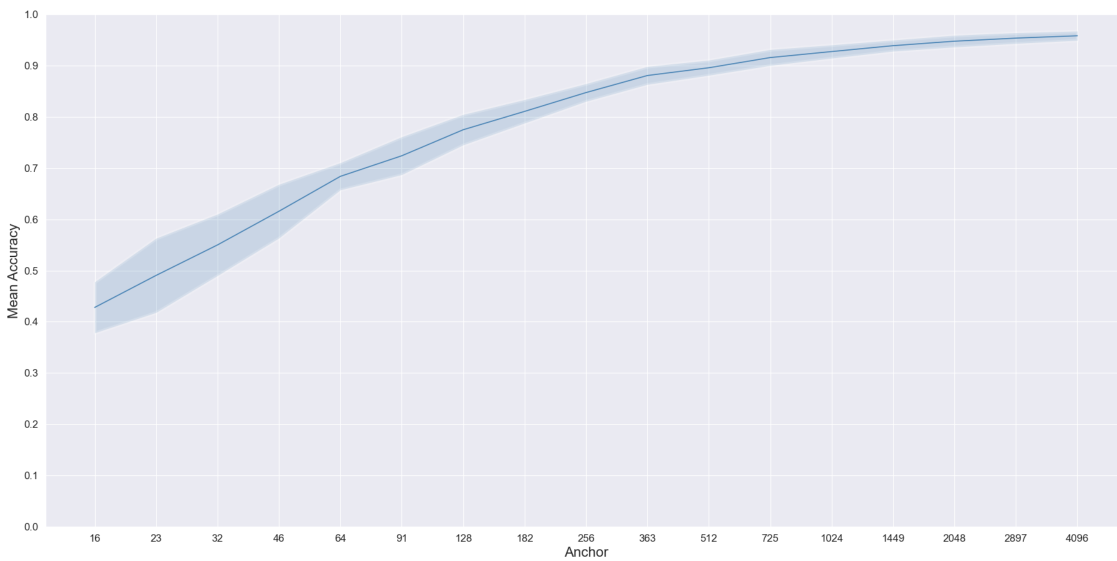


Figure 3: Example of a produced learning curve. The bold line represents the mean cross-validation accuracy, while the shaded area represents the standard deviation of the accuracy over all folds.

3.2 Datasets

To obtain a diverse range of performance landscapes, we gathered 10 different datasets that produced unique sets of learning curves.

We selected datasets from OpenML [15] based on specific attributes. These attributes are the number of instances, dimensionality, and the majority class percentage. Using filtering, we selected the most downloaded and liked datasets that do not contain missing values.

Some datasets include categorical features; however, we did not apply preprocessing, as they were already label-encoded. Lastly, we decoded any UTF-8-encoded values to enable the data to work with the gradient boosting classifier. We show the selection of datasets and their attributes in Table 2.

Dataset (ID)	#Instances	#Features	Categorical?	#Classes	Majority class %
texture (40499)	5500	41	N	11	0.091
mfeat (12)	2000	217	N	10	0.100
waveform (60)	5000	41	N	3	0.338
EEG (1471)	14980	15	N	2	0.551
banknote (1462)	1372	5	N	2	0.555
steel (1504)	1941	34	N	2	0.653
phoneme (1489)	5404	6	N	2	0.706
nomao (1486)	34465	119	Y	2	0.714
blood (1464)	748	5	N	2	0.762
churn (40701)	5000	21	Y	2	0.858

Table 2: The dataset selection with the number of instances, features, and classes. We also include the OpenML dataset IDs and the majority class percentage, and show which datasets contain categorical features.

3.3 Implementation and Hardware

Due to the computational cost of generating learning curves, a single computer system was insufficient for this task. Therefore, we performed this work using the computing resources from the Academic Leiden Interdisciplinary Cluster Environment (ALICE)[2] provided by Leiden University.

ALICE provided ample CPUs, enabling us to parallelise and speed up the creation of the learning curves. We utilised different partitions that differ in CPU hardware specifications, as documented in the ALICE wiki. Although the partitions differ in hardware, all runs used the same Python and library versions. We also applied fixed random seeds per dataset, as described in Section 3.4. These measures resulted in semi-reproducible learning curves, since slight variations may appear due to hardware differences.

We wrote the script that generated the learning curves in Python. Instances of this Python script generated a random configuration at run-time and then produced a learning curve corresponding to it. It then stores the learning curve in a JSON file corresponding to the used dataset. We can create multiple learning curves per instance, but we produced only 1 to 2 per instance because ALICE enabled us to run hundreds of instances simultaneously.

[2]ALICE wiki: https://pubappslu.atlassian.net/wiki/spaces/HPCWIKI/pages/37519361/ALICE

3.4 Methods

We generated 1,000 unique configurations of the gradient boosting classifier for each dataset. Then, we performed nested cross-validation [14] with each configuration to create the learning curves. The nested cross-validation consisted of five outer splits, each with five inner splits, yielding a total of 25 inner splits.

In this section, we will describe the process of our nested cross-validation, visually presented in Figure 4. This process largely aligns with the methods from recent work. [17] We will conclude this section by explaining how the nested cross-validation forms the learning curves.

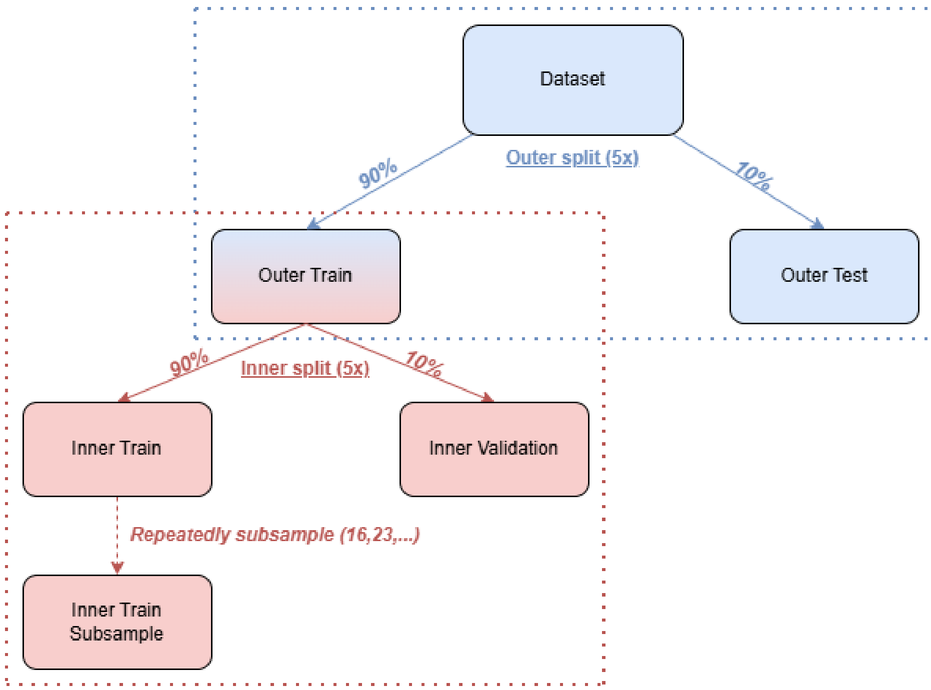


Figure 4: The process of our nested cross-validation. The blue and red areas represent the outer and inner sections, respectively.

3.4.1 Outer & Inner split

First, we randomly shuffled the dataset and split it into an outer training set and an outer test set with a 90%/10% ratio. We stratified our splits to maintain the class balance in the test set. We used a fixed seed per dataset, which allows fair comparisons between

Secondly, we performed an inner split, which divided the outer training set into an inner training set and an inner validation set. Similarly to the outer split, we stratified the split with a 90%/10% ratio. We tied the random seed of an inner split to the dataset and to the outer split iteration. This design allows each outer split to have different inner splits, while maintaining consistency between anchors and configurations.

3.4.2 Subsampling, training & validation

At this stage, we subsampled the first α instances from the inner training set to match anchor α. We then trained the classifier on this subsample and validated it on the inner validation set, yielding an accuracy score [12]. The trained classifier is also tested on the outer test fold, also yielding an accuracy score.

3.4.3 Creating the learning curves

We repeated this process for each of the 25 inner splits and performed the complete cross-validation for every anchor. Thus, for each anchor, we obtained 25 individual validation and test scores, along with the mean validation and test scores. We subsampled the anchors monotonically, meaning the subsample of an anchor is always a subset of a larger anchor.

The individual and mean scores per anchor are saved, together with the corresponding configuration, dataset, and random seed. We also save the run-time per configuration, but we do not use it in this work. Importantly, we used only the mean validation scores per anchor to build learning curves in this thesis. However, it is entirely possible to form learning curves using the mean test scores from our database, or even a combination of the two types of scores.

3.5 Visual inspection

We produced 1,000 learning curves per dataset. The learning curves show differences in behaviour between datasets, which we analyse in this section. Analysing the learning curves can help us understand the impact of these different behaviours on specific tasks, such as learning curve extrapolation (Section 4.2).

First, we created the mean learning curve for each dataset. Since each learning curve contains a configuration's cross-validation score for each anchor, we can average the scores per anchor across all learning curves to obtain the mean learning curve for a dataset. It is important to note that in this visual inspection, the cross-validation scores are equal to the mean validation errors for a configuration, rather than the mean validation accuracies.

Figure 5 shows the mean learning curves. These should roughly represent the learning curves per dataset in shape. We observe differences in the steepness and shapes of the curves. For example, the blood curves tend to be flatter than curves of other datasets, while the mfeat curves seem steep at the first anchors and flatten afterwards. Another example is the S-shaped EEG curves.

Secondly, we examine the interquartile range (IQR) of cross-validation scores to analyse the variation in the curve sets. The IQR represents the amount of variation in cross-validation scores across configurations. We calculated the IQR per anchor as the difference between the 25th and 75th percentiles of cross-validation scores for that anchor.

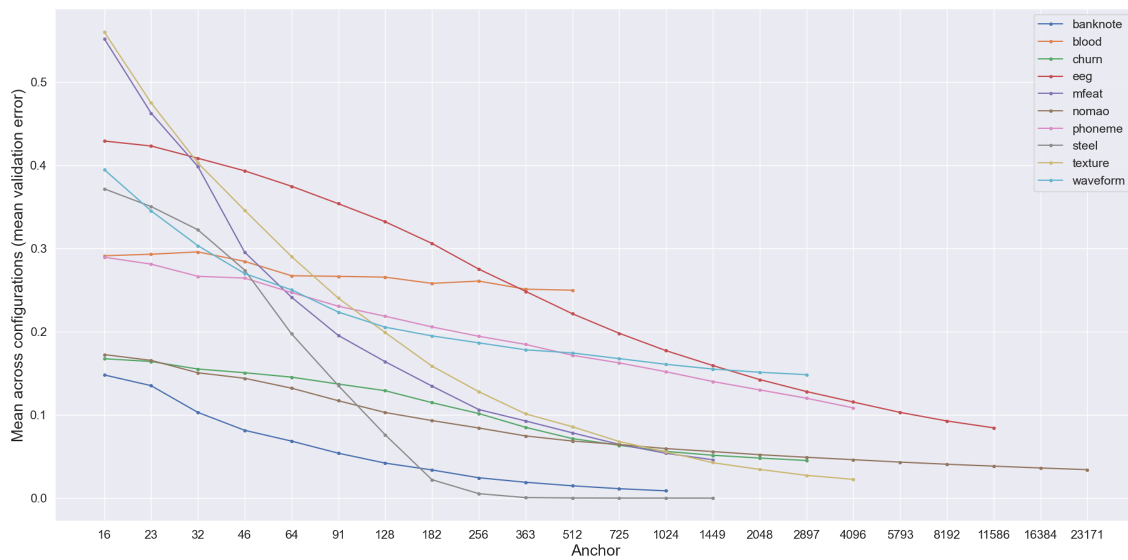


Figure 5: The mean learning curves for each dataset. The x-axis shows the anchors, and the y-axis shows the mean cross-validation scores across configurations.

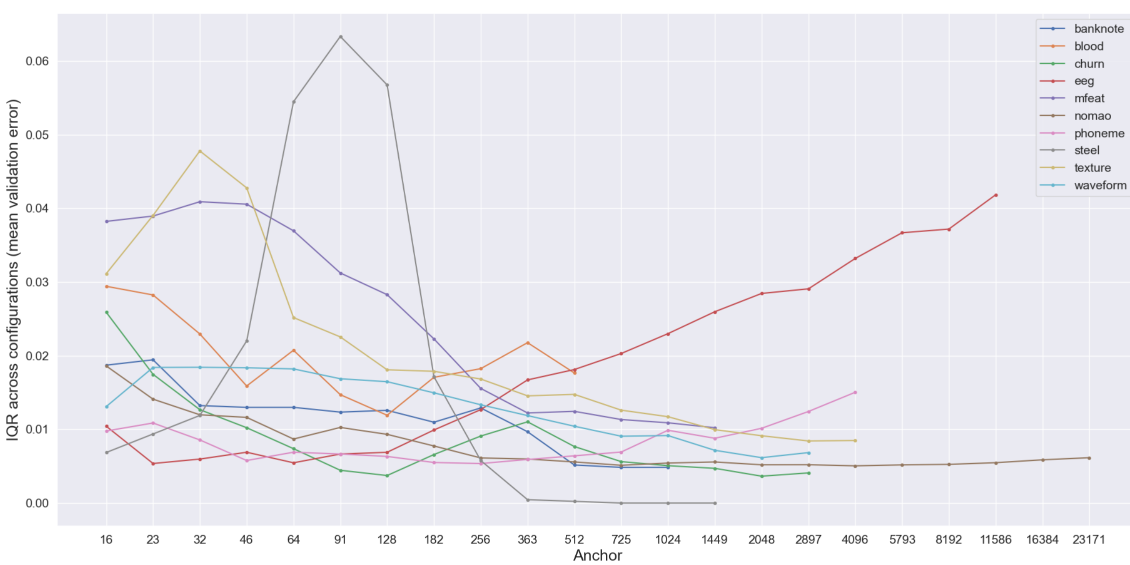
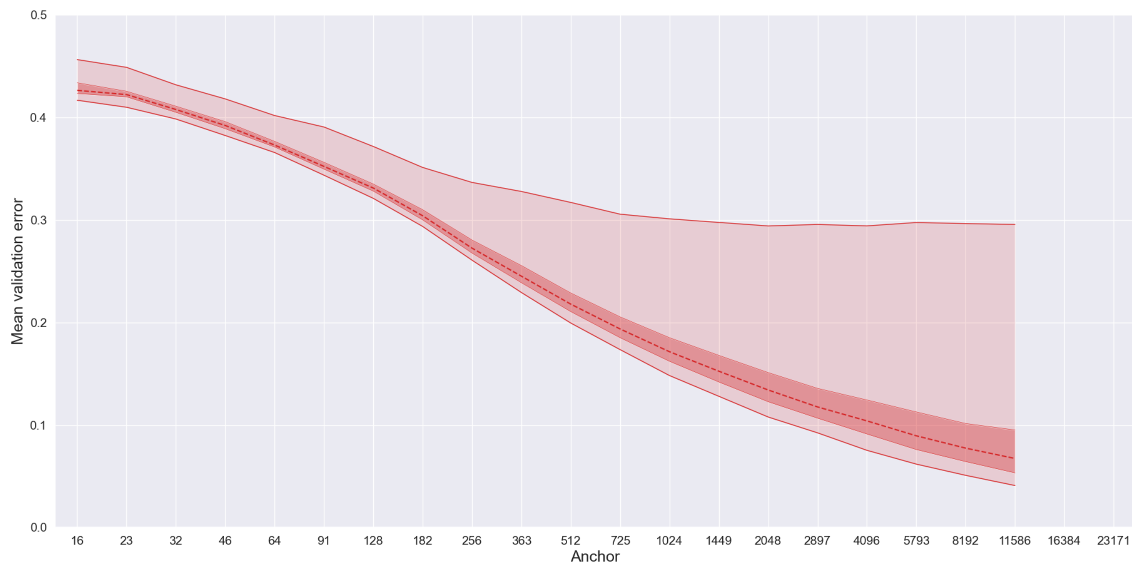
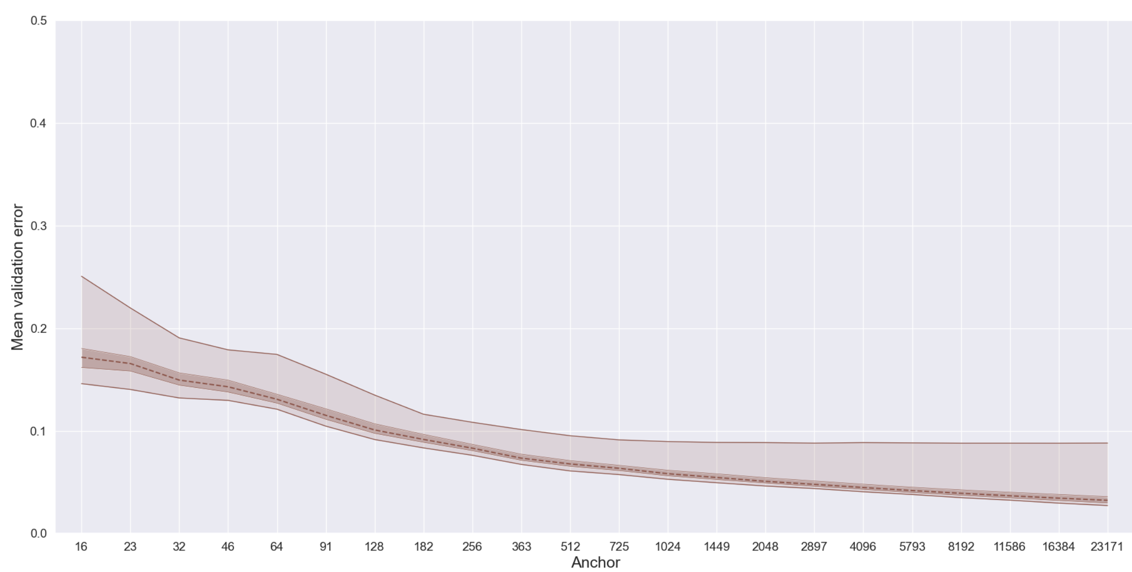


Figure 6: The IQR of the learning curves for each dataset. The x-axis shows the anchors, and the y-axis shows the IQR of the cross-validation scores.

In Figure 6, we show the IQR of the learning curves. We observe significant differences between the datasets. We show the learning curve distribution of the EEG and nomao datasets in Figure 7 to illustrate these differences. We see in Figure 7a that the variation of the EEG learning curves significantly increases in the higher anchors. In contrast, we see in Figure 7b that there is less variation in the nomao learning curves.



(a) Distribution of the EEG learning curves,



(b) Distribution of the nomao learning curves

Figure 7: The distribution of the EEG and nomao learning curves, represented by the five-number summary over the anchors. The solid lines on the outside represent the minimum and maximum; the darker-shaded area represents the IQR; and the dashed line represents the median.

4 Database usage: Curve prediction & Extrapolation

We conducted two experiments to demonstrate the use cases of our database in research and machine learning. In the first experiment, we trained surrogate models on our generated learning curves to predict learning curves for unseen hyperparameter configurations. The second experiment involves learning curve extrapolation, where we train models on partial curves and assess their extrapolation performance across different fronts.

4.1 Surrogate model performance

For this experiment, we train different regression models on learning curves and their corresponding configurations. The regressors then predict learning curves for unseen configurations. We will analyse the surrogates' performance for every dataset and also inspect the performance differences between anchors.

4.1.1 Surrogate models

Table 3 shows the selection of surrogate models. We chose a limited number of models that work well out of the box. These include two ensemble learners: random forest and gradient boosting, alongside a linear learner with ridge. We used the scikit-learn [12] implementation of these three models. This selection enabled comparison among different ensemble models, with the linear model serving as a baseline. To save time and keep the experiment simple, we chose not to optimise the models' hyperparameters and instead used their default values.

Regression model	Description	Citation
RandomForestRegressor	An ensemble learner, fits and averages the predictions of multiple decision trees.	[1]
GradientBoostingRegressor	An ensemble learner, incrementally combines multiple regression trees to form a prediction.	[4]
Ridge	A regularised linear regression model, using linear least squares as loss function with L2 regularisation	[5]

Table 3: The regression models selected to function as surrogate models. We included a short description of the models.

4.1.2 Setup

The setup for this experiment was a 5-fold cross-validation. Since we had 1,000 learning curves per dataset, splitting resulted in each fold containing 200 learning curves. We shuffled the learning curves before the split to increase fold-to-fold variation. We also implemented controlled randomness to shuffle the curves by tying the random seed to the current iteration.

Then, we trained the surrogate models on four training folds and validated them on the remaining fold. We repeated this process for five iterations, where each iteration a different fold served as the validation fold. We chose to train and validate the models on the full learning curves, using the configurations as features. The learning curves themselves served as the labels, represented as a series of scores. We performed 10 cross-validation iterations for each dataset, yielding a total of 50 distinct validation folds per dataset.

4.1.3 Performance Metric

For each predicted learning curve in the validation set, we calculated the prediction error per anchor and also the mean error over all anchors. We considered the root mean square error (RMSE) and the mean absolute error (MAE) to evaluate the models. The RMSE emphasises lower error values more than the MAE. We decided that the RMSE was a better fit, since the surrogates produced relatively small error values.

4.1.4 Results

In this section, we discuss the results of this experiment. We first compare the performance of the surrogates per dataset. Then, we compare the surrogates' performance in predicting different anchors of the learning curves.

Figure 8 shows a violin plot of the surrogates' performance per dataset. Each violin represents the distribution of RMSE scores across 50 validation folds for a specific surrogate-dataset combination. We observe that the random forest and gradient boosting surrogates outperform the ridge surrogate for all datasets. Furthermore, we note that the gradient boosting surrogate slightly outperforms random forest across most datasets while exhibiting less variance.

Interestingly, the performance gap between surrogates is more pronounced for datasets such as EEG, texture, and mfeat. This disparity may stem from the learning curves of these datasets, which display more variance than those of other datasets, as illustrated in Figure 6.

Lastly, the violins for the steel dataset exhibit greater performance variance than those for other datasets. We can again attribute this to the learning curves of this dataset, which display more variance in the middle anchors and near-zero variance in the higher anchors, as shown in Figure 6. This difference in variance likely accounts for the variation in the surrogates' performance.

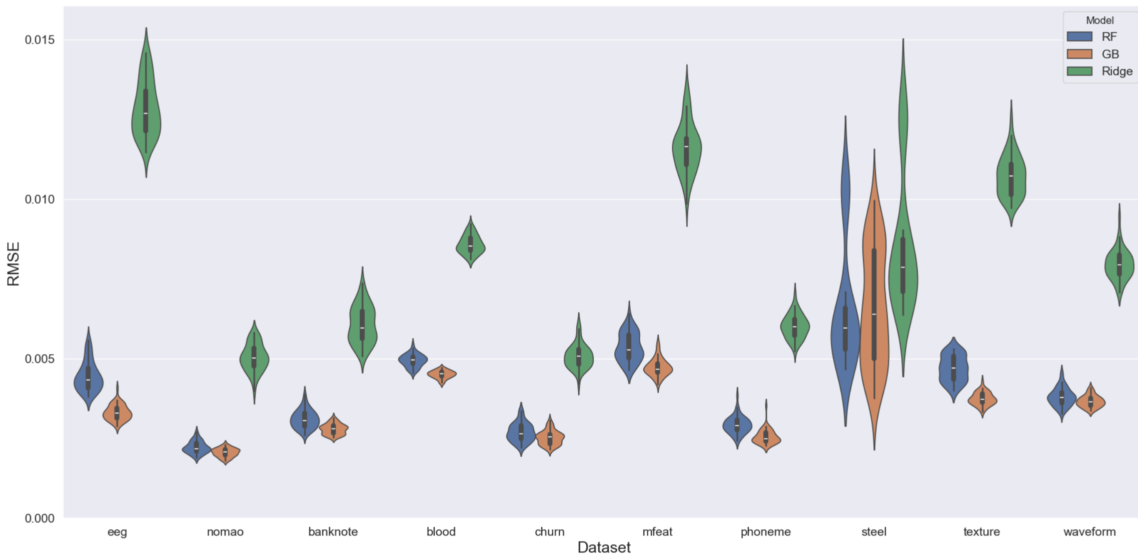


Figure 8: Violin plot of the models' RMSE scores per dataset. Each violin contains a box-and-whisker plot that displays the distribution of RMSE.

In Figure 9, we show the surrogates' performance for different anchors. While we generally observe higher RMSEs in the early anchors and lower ones in the later ones, we also see inconsistent relationships between RMSE and anchor size. Instead, it is more likely that performance is affected by the variance in the learning curves, especially since we use only 10 datasets. If we compare Figure 9 with Figure 6, we can see that the variance of the steel and EEG curves correlates with the RMSE increase in Figure 9 around the 91 and 2048 anchors, respectively.

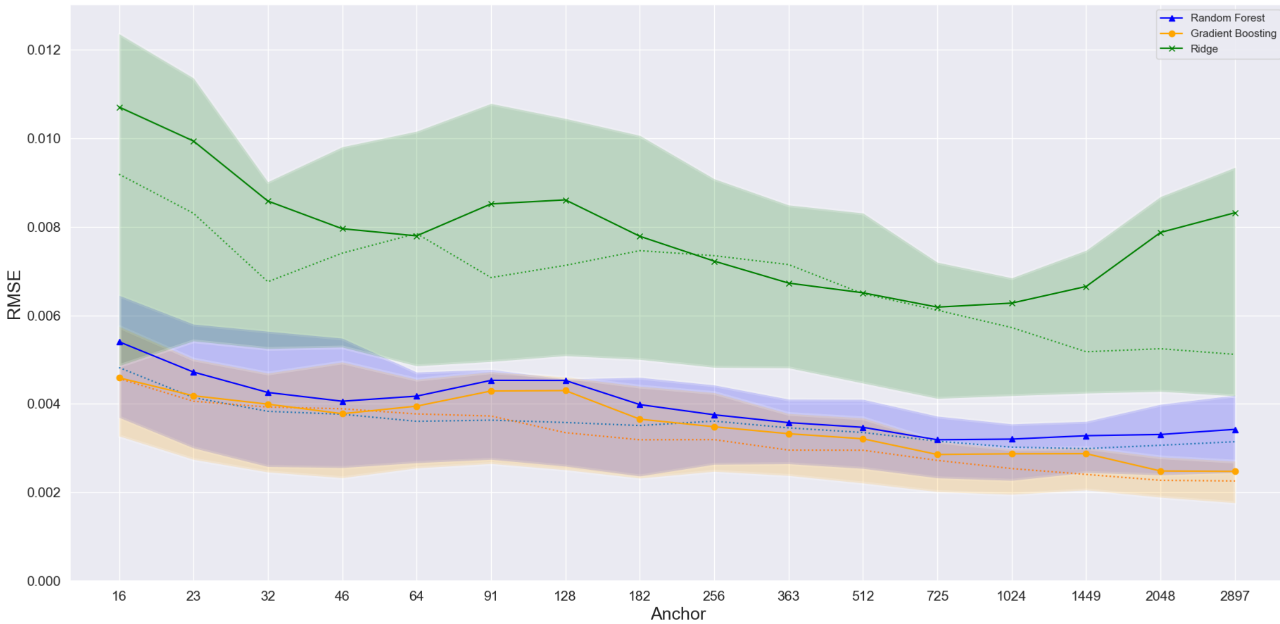
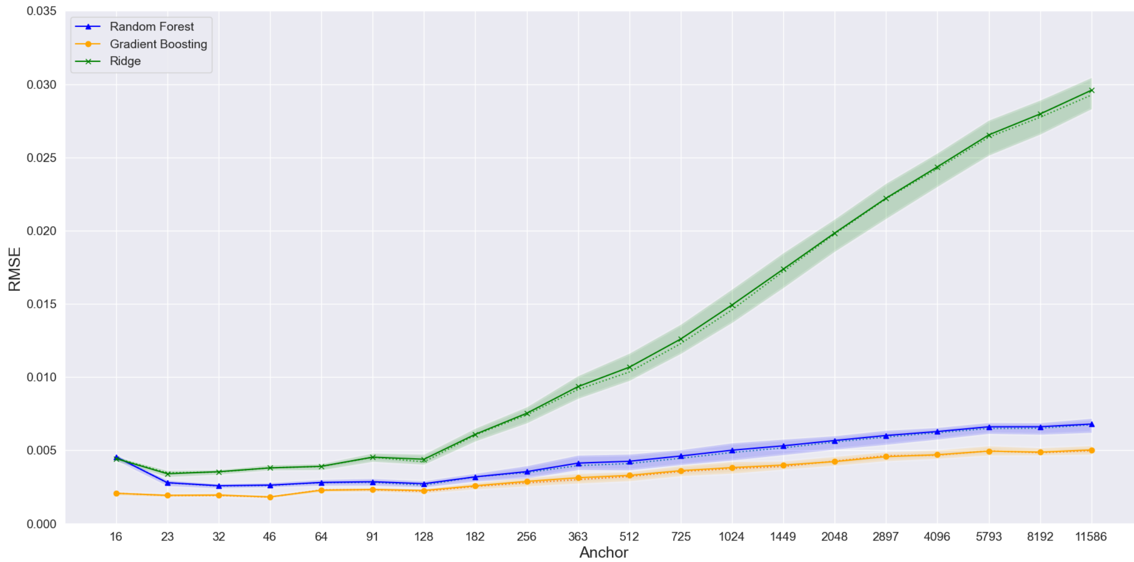


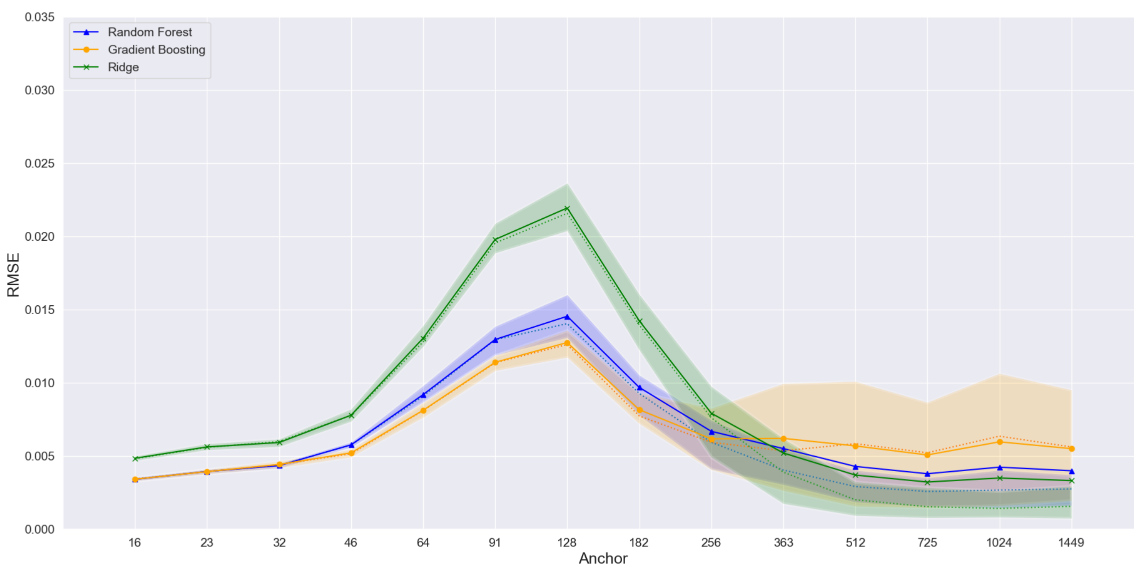
Figure 9: The performance of the surrogate models for each anchor. The solid lines represent the mean RMSE over all datasets, while the dotted lines indicate the median RMSE over all datasets. To ensure a fair comparison of performance among the anchors, we include only the anchors that appear in at least half of the datasets.

To provide more insight, we show the surrogate model's performance across anchors for the EEG and steel datasets in Figure 10. The aforementioned correlation between the variance of the learning curves and the performance of the surrogates is clearly reflected again in Figures 6 and 10. Interestingly, ridge seemed to be impacted more by this correlation than the other two surrogates.

In summary, there is strong evidence that ensemble learners, such as random forest and gradient boosting, are better at predicting learning curves for unseen configurations than a linear model like ridge. The models' performance also seemed to depend mainly on the amount of variance in the learning curves, which can vary across datasets and anchors.



(a) Surrogate models' mean and median RMSE per anchor for the EEG dataset.



(b) Surrogate models' mean and median RMSE per anchor for the steel dataset.

Figure 10: The surrogate models' performance per anchor for the EEG and steel datasets. The solid lines represent the mean RMSE, while the dotted lines indicate the median RMSE. The x-axis labels are inconsistent across the figures because the datasets have different numbers of anchors.

4.2 Learning curve extrapolation

In this experiment, we assess our models' ability to extrapolate learning curves. The models extrapolate partial curves of different sizes to a set target anchor. In short, the goal is to compare the models' performance in learning curve extrapolation while observing the impact of different partial curve sizes.

4.2.1 Models

We picked two different types of models. Firstly, we chose the mmf4 model to represent learning curve models. This choice aligns with recent work [11], where this model performed best at their extrapolation task. Secondly, we chose scikit-learn's [12] random forest regression model to contrast mmf4.

We did not perform hyperparameter optimisation because we did not want the models to have access to the target anchor's scores. We set the model hyperparameters to their default values for simplicity.

4.2.2 Process

In this experiment, we define the partial curves as the learning curves that the models extrapolate to a target anchor. The target anchor is the anchor for which the models predict a score.

At the start, the partial curve only consists of the first anchor '16'. Then we iteratively add the following anchors until the partial curve includes all the anchors preceding the target anchor. We chose '2897' as the target anchor so that at least half of the datasets can extrapolate to it. Table 4 shows the selection of datasets for this experiment.

Dataset (ID)	#Instances	#Features	Categorical?	#Classes	Majority class %
texture (40499)	5500	41	N	11	0.091
waveform (60)	5000	41	N	3	0.338
eeg (1471)	14980	15	N	2	0.551
phoneme (1489)	5404	6	N	2	0.706
nomao (1486)	34465	119	Y	2	0.714
churn (40701)	5000	21	Y	2	0.858

Table 4: The dataset selection for this experiment. We include the OpenML dataset ID [15], the majority class percentage, and the numbers of instances, features, and classes. We also show which datasets contain categorical features.

We trained the random forest model on all the partial curves at once. The model would then predict the test anchor score for each curve. We fit the mmf4 model on each partial curve separately, and then extrapolated it to the target anchor. To fit the model, we were allowed to perform five random restarts if the training MSE exceeded 100, and we used the best parameters based on the training

MSE. For this, we make use of a fixed random seed. This mmf4 setup mostly aligns with recent work [11].

We chose this setup because we wanted the models to use information only from the partial curves. In different contexts, where we would train models on another set of full curves, for example, it would be unsuited to compare regressors like random forest to mmf4, which is not inherently able to utilise other curves as information.

We compare the two models with a baseline called 'Last'. This baseline aligns with recent work [7, 11]. To create this baseline, we used the same datasets and learning curves to iteratively grow partial curves, with the target anchor set to 2897. The baseline is then built iteratively by using the last anchor in the partial curves as the prediction. The target anchor scores again serve as the true scores.

4.2.3 Performance Metrics

We decided to use the RMSE as the measurement metric for similar reasons as the first experiment. We also measured the ranking performance with a metric called Top-100 precision error. It represents the set difference between the top 100 scores in the actual ranking and those in the ranking of the model's predictions.

4.2.4 Results

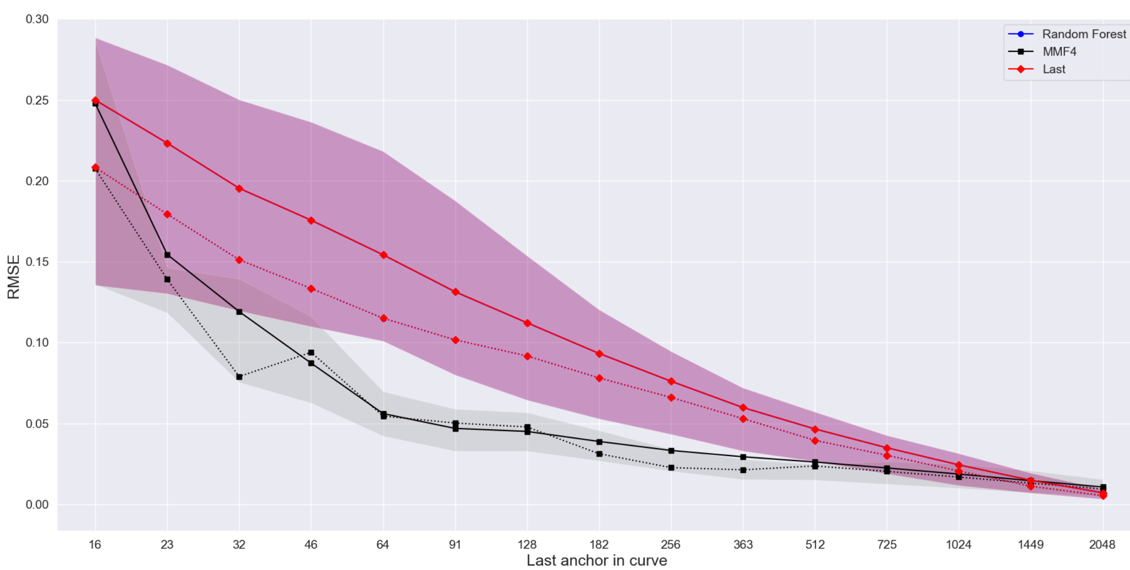


Figure 11: The learning curve extrapolation performance in RMSE. The solid lines represent the mean RMSE over the datasets, while the dotted lines represent the median RMSE over the datasets. The shaded area represents the interquartile range. The x-axis shows the last anchor in the partial curves, with the target anchor set to 2897.

In Figure 11, we display the extrapolation performance in RMSE for different partial curve cut-offs. We notice that the performances of the random forest and Last models are nearly identical. Since the random forest model is trained only on partial curves and lacks information about the target anchor, we theorise that the tree-based design leads it to extrapolate toward the score of the highest anchor it has seen, which would be the last anchor in the partial curve.

On the other hand, we observe that the mmf4 model clearly outperforms the other models, particularly after the first few anchors. However, for the last two anchors, it struggles to do the same. Since the learning curves tend to flatten out at these anchors, it becomes hard to outperform Last and, consequently, random forest.

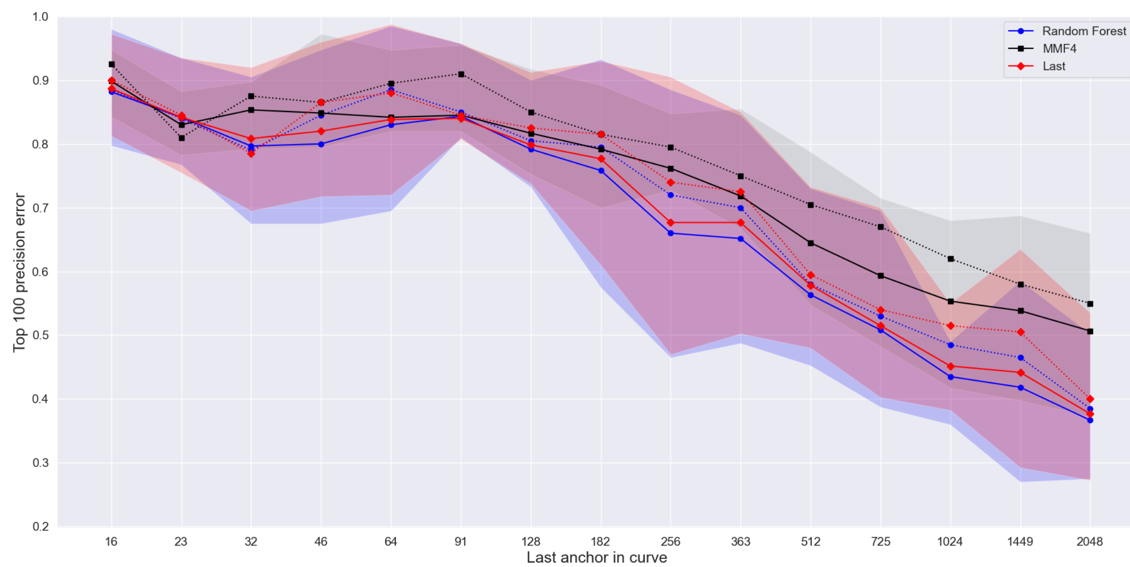


Figure 12: The learning curve extrapolation performance in Top-100 precision error. The solid lines represent the mean error across datasets, while the dotted lines represent the median error across datasets. The shaded area represents the interquartile range. The x-axis shows the last anchor in the partial curves, with the target anchor set to 2897.

In Figure 12, we show the extrapolation ranking performance for different cut-offs in the partial curves. This metric measures their ability to predict the ranking of the actual scores. We observe that random forest slightly outperforms the baseline and clearly outperforms mmf4, especially in the higher anchors. The random forest model likely benefits from being trained on the learning curves and having access to their configurations, which mmf4 and Last miss out on.

To summarise, we observed how the models perform at extrapolating in different aspects. mmf4 performed best at extrapolating exact scores for unseen anchors, whereas random forest struggled in this setup. However, we observed that random forest performed best at predicting rankings of scores for unseen anchors, due to its ability to leverage learning curve configurations for predictions.

5 Conclusions

In this section, we summarise our thesis, state our conclusion, and discuss the limitations of our research. We also address the possibilities for future research.

5.1 Summary and Concluding Remarks

This thesis produced a new learning curve database, focusing on curves of various unique hyperparameter configurations of the gradient boosting algorithm. These curves were briefly analysed and then used in two experiments, demonstrating how this database can support machine learning tasks.

We wrote a Python script to produce the learning curves and ran it on the ALICE computer cluster. In the script, we performed cross-validation with the gradient boosting algorithm for different hyperparameter configurations. We used various amounts of training data (anchors), for which the validation and test performance were produced and stored. Instances of our database consist of these scores, plus information such as the used random seed, run time, and the corresponding configuration and database. The validation and test scores can both serve to create learning curves, but for our thesis, we built them only using the mean validation scores per anchor.

By visually inspecting the learning curves, we found differences in curve behaviour across datasets, including characteristics like curve shapes and variation. We found that the curve variation can differ significantly between datasets and also among anchors within each dataset. These observations provided insights into the wide range of learning curve behaviours and offered context for the two experiments that utilised these curves.

These two experiments aimed to demonstrate how our database supports machine learning research tasks. In the first experiment, we predicted learning curves for unseen hyperparameter configurations with surrogate models. We trained these surrogates on full learning curves and then performed regression in a cross-validation setup. The results showed a diverse range of surrogate performance for multiple sets of learning curves. We produced a violin plot of their error rate and found that the random forest and gradient boosting surrogates outperformed ridge for all datasets. We also observed that across most datasets, gradient boosting slightly outperformed random forest in terms of error and consistency.

We also plotted the surrogates' error rate for different anchors. We analysed the mean, median, and interquartile range to study their performance. We generally observed an inconsistent relationship between error rate and anchor size. However, we speculated that the variance in performance correlated with the variance of the learning curves. To further investigate this, we inspect the surrogates' performance for two different sets of learning curves. We observed the aforementioned correlation again in these results and therefore concluded that the performance of these models likely depends on the variance of the learning curves set.

The second experiment entailed learning curve extrapolation, using different types of models. We decided to compare a learning curve model with a regressor, represented by the mmf4 model and the random forest regressor, respectively. While the extrapolation process of these models differed in some aspects, they both extrapolated the same set of partial curves to a fixed target anchor. We iteratively grew the partial curves, initially containing only the first anchor, to contain the anchors up to the target anchor, and repeated the extrapolation process for each iteration. We measured the extrapolation performance for different partial curve sizes. We also include a baseline called 'Last', which essentially uses the last anchor in the partial curve as the prediction.

In terms of accuracy, we found a significant overlap between the random forest regressor and the Last baseline. Our extrapolation setup for the random forest regressor likely caused this overlap. Nonetheless, both were outperformed by mmf4 at most partial curve sizes, demonstrating its ability to extrapolate learning curves with more accuracy than the other models.

Still, we observed different results when we measured the performance in terms of ranking. The random forest regressor generally outperformed both mmf4 and the Last baseline on this front. Random forest utilised all the partial curves and their configurations as information, which likely provided a significant advantage. Overall, we concluded that random forest outperformed in ranking precision, while mmf4 seemed to outperform in extrapolation accuracy.

In conclusion, this thesis successfully achieved its goal of creating a new learning curve database and demonstrating its value for machine learning research through various experiments. We highlighted multiple characteristics of our learning curves and discussed the correlation between learning curve variation and model performance in our experiments. The findings of our experiments demonstrate that the database can effectively support a range of machine learning tasks and may also serve as a source of inspiration for future research.

5.2 Limitations & Future Research

The database has much room for improvement. The majority of datasets lack categorical variables or only contain two classes. Including more datasets with different attributes would make the database more robust for use in research. A minor flaw lies in the database's reproducibility. While we included measures such as fixed random seeds and documented the specific hardware and software used to create the database, slight differences in the curves may appear for other hardware and operating systems.

A couple of design choices in the experiments were sub-optimal, including the sparse selection of models in both experiments and the absence of hyperparameter optimisation in the first experiment. We also did not perform any statistical analysis to answer specific research questions. Instead, we aimed to demonstrate the database's research utility through our experiments.

Besides addressing these limitations, future research would naturally focus on what this thesis did not cover. Most importantly, the database could support future research on shapes and behaviours of learning curves, which we barely touched upon in our work. Future work could also include more extensive research using our database for tasks related to hyperparameter optimisation or model selection. Lastly, we did not utilise the run-times from our learning curves in our experiments, but they could serve as a resource for future research.

References

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] Charles Edison Tripp, Jordan Perr-Sauer, Lucas Hayne, Monte Lunacek, and Jamil Gafur. The butter zone: An empirical study of training dynamics in fully connected neural networks. *arXiv preprint arXiv:2207.12547*, 2022.

[3] Katharina Eggensperger, Marius Lindauer, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Efficient benchmarking of algorithm configurators via model-based surrogates. *Machine Learning*, 107(1):15–41, 2018.

[4] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[5] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[6] Kevin G. Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016*, volume 51 of *JMLR Workshop and Conference Proceedings*, pages 240–248. JMLR.org, 2016.

[7] Lionel Kielhöfer, Felix Mohr, and Jan N. van Rijn. Learning curve extrapolation methods across extrapolation settings. In *Advances in Intelligent Data Analysis XXII - 22nd International Symposium on Intelligent Data Analysis, IDA 2024*, volume 14642 of *Lecture Notes in Computer Science*, pages 145–157. Springer, 2024.

[8] Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18:185:1–185:52, 2017.

[9] Luke Metz, Niru Maheswaranathan, Ruoxi Sun, C Daniel Freeman, Ben Poole, and Jascha Sohl-Dickstein. Using a thousand optimization tasks to learn hyperparameter search strategies. *arXiv preprint arXiv:2002.11887*, 2020.

[10] Felix Mohr and Jan N. van Rijn. Learning curves for decision making in supervised machine learning: a survey. *Machine Learning*, 113(11):8371–8425, 2024.

[11] 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*, pages 3–19, 2022.

[12] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] Janek Thomas, Stefan Coors, and Bernd Bischl. Automatic gradient boosting. *arXiv preprint arXiv:1807.03873*, 2018.

[14] Hado van Hasselt. Estimating the maximum expected value: An analysis of (nested) cross validation and the maximum sample average. *arXiv preprint arXiv:1302.7175*, 2013.

[15] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

[16] Tom J. Viering and Marco Loog. The shape of learning curves: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7799–7819, 2023.

[17] Cheng Yan, Felix Mohr, and Tom J. Viering. LCDB 1.1: A database illustrating learning curves are more ill-behaved than previously thought. *arXiv preprint arXiv:2505.15657*, 2025.

[18] Lucas Zimmer, Marius Lindauer, and Frank Hutter. Auto-Pytorch: Multi-fidelity metalearning for efficient and robust autoDL. *IEEE transactions on pattern analysis and machine intelligence*, 43(9):3079–3090, 2021.