Learning Learning Curves

Turan, O. Taylan; Tax, David M.J.; Viering, Tom J.; Loog, Marco

DOI
10.1007/s10044-024-01394-6

Publication date
2025

Document Version
Final published version

Published in
Pattern Analysis and Applications

Pattern Analysis and Applications (2025) 28:15
https://doi.org/10.1007/s10044-024-01394-6

ORIGINAL ARTICLE

Learning Learning Curves

O. Taylan Turan[1] · David M. J. Tax[1] · Tom J. Viering[1] · Marco Loog[1,2]

Received: 29 August 2024 / Accepted: 6 December 2024

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

Keywords Meta-learning · Data-driven modeling · Learning curves · Extrapolation · Kernel methods · functional data analysis

1 Introduction

A learning curve shows the generalization performance of a learner as a function of the training set size. This should not be confused with training curves. Both curves find various applications in machine learning pipelines. Tuning hyperparameters, selecting models, and assessing whether adding more data benefits a learner, for instance, are some of the applications [1]. However, training curves are used to track model performance for a single learning problem during loss optimization, whereas the learning curve tracks the average model performance over different learning problems from the same dataset. In other words, one obtains the training curve at no cost while training the model. However, obtaining learning curves require training the model multiple times with different subsets of the dataset, making it computationally taxing to obtain.

The performance of a learner for increasing training set sizes can be predicted by extrapolating the learning curves.

✉ O. Taylan Turan
o.t.turan@tudelft.nl

David M. J. Tax
d.m.j.tax@tudelft.nl

Tom J. Viering
t.j.viering@tudelft.nl

Marco Loog
marco.loog@ru.nl

[1] Intelligent Systems, Delft University of Technology, Van Mourik Broekmanweg, Delft 2628XE, South Holland, The Netherlands

[2] Institute for Computing and Information Sciences, Radboud University, Toernooiveld, Nijmegen 6525EC, Gelderland, The Netherlands

Published online: 03 January 2025

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

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

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

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

To break free from limitations of parametric functions, we assume the existence of a database of learning curves, as shown in Fig. 1a. The goal is to extrapolate an unknown targeted learning curve to get a generalization performance at a specific training set size, as depicted in Fig. 1b, without using any parametric models.

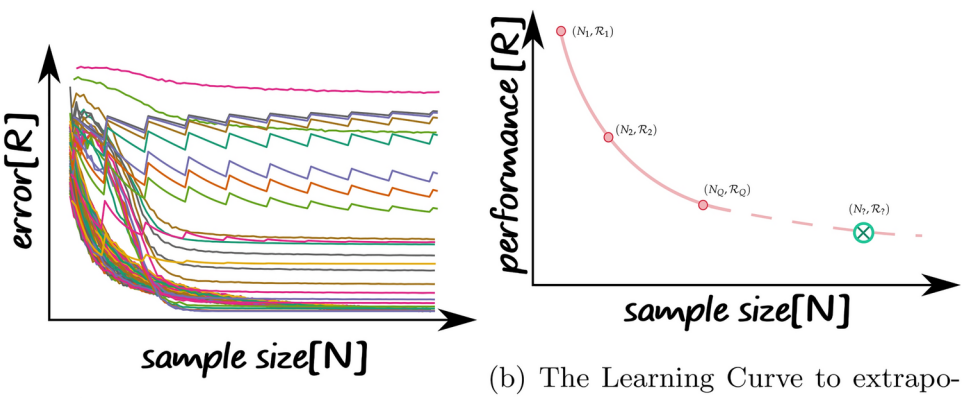
A similar setting is considered in [7], where, a non-parametric, data-driven pipeline is introduced that uses a similar setting where a learning curve database is available. However, their task is to perform a binary model selection for classification problems at unseen sample sizes. We propose a pipeline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

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

This paper is organized as follows: In Sect. 2, we provide a brief background on learning curves. Section 3 presents common methods for extrapolating learning curves, along with our proposed methodology. We then detail our experimental setup in Sect. 4. Section 5 presents our results, followed by the main conclusions in Sect. 6. Before giving our learning curve definition in Sect. 2, we should mention that this paper deviates from the works proposed on training curve extrapolation [10–16], simply because of the inherent differences between learning and training curves.

2 Background on the learning curves

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



(a) Learning Curve Database

(b) The Learning Curve to extrapolate.

Fig. 1 Learning curve plots show generalization performance vs sample size. In b: initial points observed marked with pink, and the desired training set size marked with green for which we would like to get the generalization performance

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

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

$$\mathcal{R}(h) = \int \mathcal{L}(y, \hat{y}) \mathcal{P}(x, y) dx dy \tag{1}$$

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

3 Extrapolating learning curves

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

3.1 Parametric curve-fitting

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

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

According to [1, 2], many researchers use power law and exponential parametric models for f. To account for behaviour changes for different regions of the learning curves, more complex parametric formulations are proposed in [5, 18]. All the parametric models proposed are non-linear functions of $\boldsymbol{\theta}$. This makes parametric curve-fitting initialization dependent. When this is the case, Eq. 2 does not have a closed form solution and is non-convex.

3.2 Semi-Parametric Kernel Ridge (*SPKR*)

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

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

$$\hat{\tilde{f}} \in \arg\min_{\tilde{f} \in \mathbb{H}} \mathcal{L}(\tilde{f}, \boldsymbol{\mathcal{R}}) + \Omega(||f||_{\mathbb{H}}). \tag{3}$$

In this equation, $\mathbb{H}$ is the Reproducing Hilbert Space and $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{Q \times 1}$ is all the labels of our training set. According to the *Semi-parametric Representer Theorem* [19], the solution to Eq. 3 has the form: $\hat{\tilde{f}}(\cdot) = \sum_i^Q \alpha_i K(\cdot, N_i) + \sum_j^M \beta_j \psi_j(\cdot)$, where K is any Mercer kernel. If we assume $\Omega(||f||_{\mathbb{H}}) := \lambda ||f||_{\mathbb{H}}^2$ and the squared error as loss function $\mathcal{L} = \sum_{i=1}^Q (\mathcal{R}_i - \hat{f}(N_i, \boldsymbol{\theta}))^2$, the convex optimization problem has a unique solution. The optimal parameters are represented by $\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{Q \times 1}$, and $\hat{\boldsymbol{\beta}} \in \mathbb{R}^{M \times 1}$. Furthermore, when $\mathrm{K} \in \mathbb{R}^{Q \times Q}$, $\boldsymbol{\psi} \in \mathbb{R}^{Q \times M}$ are the kernel matrix, and the additional information coming from the available learning curves, respectively, this unique solution can be obtained as:

$$\hat{\mathrm{w}} = (\mathrm{A}^{\mathrm{T}}\mathrm{A} + \lambda \mathrm{B}^{\mathrm{T}})^{-1} \mathrm{A}^{\mathrm{T}} \boldsymbol{\mathcal{R}}. \tag{4}$$

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

$\mathrm{B} := [\mathrm{I}, 0; 0, 0] \in \mathbb{R}^{(Q+M)\times(Q+M)}$ is the regularization applied to f.

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

One of the most obvious choices is to use all the similar curves in the database, but note that there is an inverse operation in Eq. 4 with the complexity $\mathcal{O}((Q+M)^3)$. This results in a computational bottleneck when the relevant part of the database is large. To keep the computations feasible, a small set of curves representing the biggest modes of variations is derived using principal component analysis (*PCA*) [20]. Next to reducing the computational cost, it also prevents overfitting to the learning curve database. Our learning curve database consists of functionals; hence we use functional principal component analysis (*FPCA*) to extract the most important modes of variations in the learning curve database.

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

$$\int v(s,t)\xi(t)dt = \rho\xi(s). \tag{5}$$

This eigenvalue problem is satisfied for each eigenfunction ξ with the corresponding eigenvalue ρ. Eigenfunctions, in this case, represent the modes of variation in the database. We choose the first M of these eigenfunctions with the largest eigenvalues ($\{\psi_p\}_{p=1}^{M} = \{\xi_p\}_{p=1}^{M}$) to be used in the *SPKR*. Since the principal components describe the principal variations around the mean $\mu_{\mathcal{C}}$ of the relevant part of the database, we also separately include it in our model definition. Thus, our final proposed model for learning curve extrapolation takes the form $\tilde{f} = f + h + \mu_{\mathcal{C}}$. Similar to Eq. 4, the optimal solution with the addition of the database mean at the training points $\bar{\boldsymbol{\mathcal{C}}} := \{\mu_{\mathcal{C}}(N_i)\}_{i=1}^{Q}$ can be obtained as:

$$\hat{\mathrm{w}} = (\mathrm{A}^{\mathrm{T}}\mathrm{A} + \lambda\mathrm{B}^{\mathrm{T}})^{-1}\mathrm{A}^{\mathrm{T}}(\boldsymbol{\mathcal{R}} - \bar{\boldsymbol{\mathcal{C}}}). \tag{6}$$

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

4 Experimental setup

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

4.1 Learning curve database

The true risk in Eq. 1 can only be computed when we have access to the exact joint distribution $\mathcal{P}(x, y)$ and when the integral is tractable. In practice, however, we only have access to a finite sample drawn from $\mathcal{P}(x, y)$. To estimate a learning curve, we choose to repeatedly select a subset of size N from the available dataset, using the remaining data to compute the test error [2]. This process is repeated 100 times, and the resulting errors are averaged to get an estimate for Eq. 1. We apply this procedure for each training set size in the range $N \in [2, 100]$ to generate one complete learning curve.

Our database is comprised of $11,240$ learning curves, of which 4840 are classification problems and 6400 are regression problems. This collection of learning curves involve curves that are known to exhibit non-monotonic behaviour (*i.e.* sawing-type [8] and special high dimensional Gaussian Process model learning curves [22], dipping phenomenon [9]) as well as monotone learning curves.

Classification models are selected to be Linear Discriminant (*LDA*), Quadratic Discriminant (*QDA*), Nearest Mean (*NMC*) and Nearest Neighbor (*NNC*) classifiers. Learning

Algorithm 1 Extrapolation with SPKR on learning curves

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

curves of these classifiers with range of hyperparameters are created for the Banana (*BAN*), Gaussian (*GAU*) and Dipping [9] (*RDIP-DDIP*) datasets. Different variants of the Banana and Gaussian datasets are obtained by varying the separation of the two classes. Moreover, For the Dipping dataset dimension of the problem (*DDIP*) and the radius of the outer class (*RDIP*) increased to create various datasets.

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

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

4.2 Learning curve extrapolation

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

4.2.1 Baselines

We choose the parametric curve-fitting baselines as *WBL4* [23] and *MMF4* [24], with parametric functions given in Eqs. 7 and 8 respectively.

$$f_{\text{wbl4}}(N, \boldsymbol{\theta}) = -\theta_1 \exp(-\theta_2 (N)^{\theta_3}) + \theta_4 \tag{7}$$

$$f_{\text{mmf4}}(N, \boldsymbol{\theta}) = (\theta_1\theta_2 + \theta_3 \cdot (N)^{\theta_4})/(\theta_2 + (N)^{\theta_4}) \tag{8}$$

These two parametric functions are reported to perform the best for extrapolating learning curves in [1] for another learning curve database. We use *LBFGS* [25] to solve Eq. 2. Since gradient approximation (*i.e.* finite difference) results are reported to be unstable for optimization, we use the exact gradients for both parametric models [14]. The optimization is repeated 100 times where the initial parameters are drawn from a normal distribution with zero mean and unit variance. For each observed curve we do a 80–20% random train-validation split on the observed data of that curve. The best performing parameter configuration on the validation set is used to initialize the fitting procedure again with the 100% of the observed data. Note that the baseline models do not have access to the learning curve database by construction.

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

$$f_{\text{bnsl}_k}(N, \boldsymbol{\theta}) = \theta_1 + (\theta_2 N^{-\theta_3}) \prod_{i=1}^{k} (1 + (N/\theta_{1_i})^{1/\theta_{2_i}})^{-\theta_{3_i}\theta_{2_i}}. \tag{9}$$

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

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

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

the scaled curves are averaged to predict generalization performance at N_{target} which is given by:

$$f_{\text{mds}}(N, \mathbb{C}, k) = \frac{1}{k} \sum_{i \in \mathbb{C}_k} s_i \mathcal{C}_i(N). \tag{10}$$

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

4.2.2 Fitting semi-parametric kernel ridge

Our method has several hyperparameters, including the regularization parameter (λ), kernel parameters (*e.g.* length-scale of the kernel), the number of *FPCA* components, similarity measure, and set the number of curves for selecting the relevant part of the database. We choose similarity measure as cosine similarity and the number of curves to be selected from the database as 100. Then, Nadaraya-Watson smoothing [27] with default values is applied to make the eigenfunctions smoother. In this work, we select M *FPCA* components which represent 95% of the selected subset of the database.

For our method we optimize only for the length-scale of a Gaussian Kernel length scale in the range $\gamma \in [10^0, 10^2]$ and a regularization parameter $\lambda \in [10^0, 10^1]$. Similar to the approach used for baselines, an 80%-20% train-validation

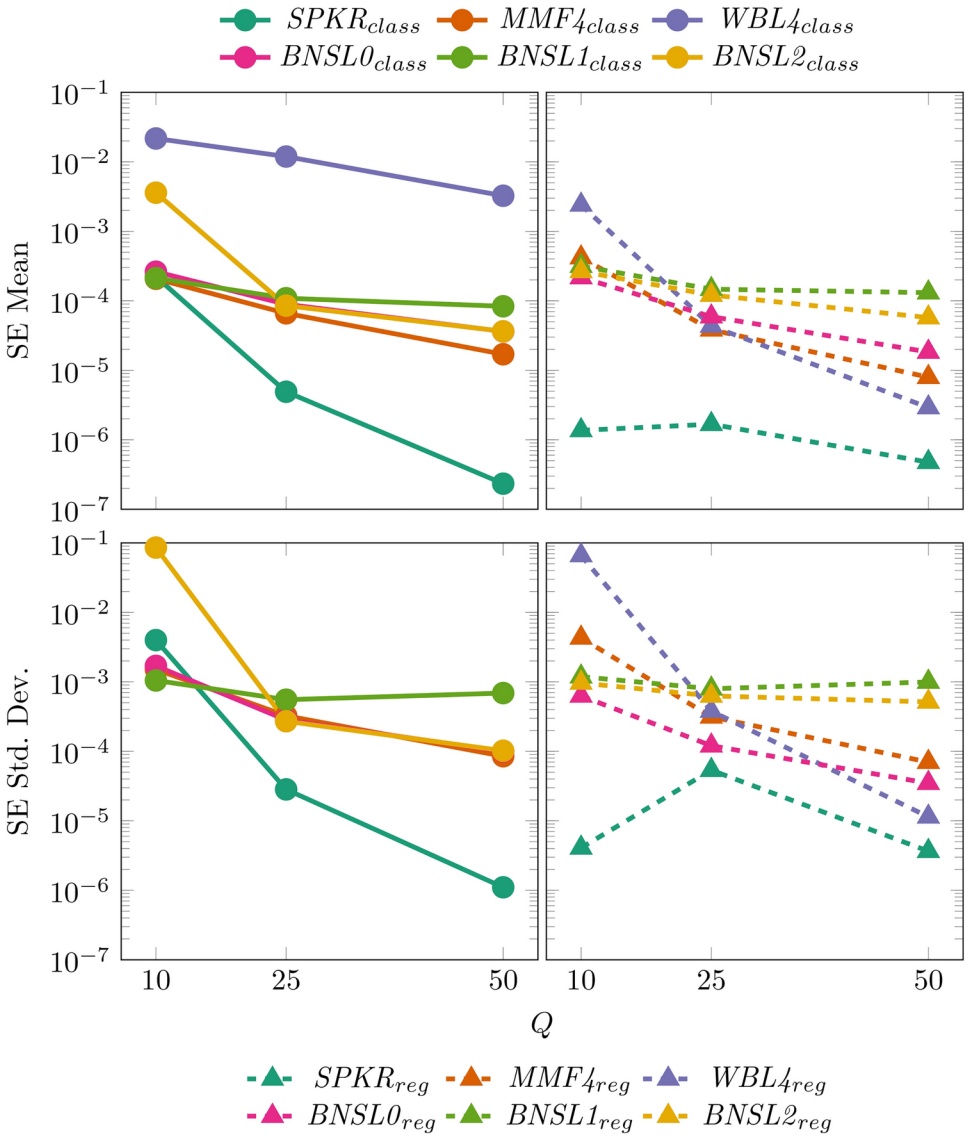


Fig. 2 Mean and standard deviation of the extrapolation squared errors (SE) for varying training points. Classification and regression learning curve results are plotted with solid and dashed lines respectively

split of the targeted curve is followed by a grid search for hyper-parameter optimization. Each hyper-parameter range is divided into 20 evenly spaced values in logarithmic space. The hyper-parameter configuration that yields the lowest validation error is selected for training with the full observed data of the partially observed curve. The resulting model is evaluated for extrapolation at $N = 100$. This process is carried our for each curve to be predicted.

5 Results and discussion

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

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

5.1 Extrapolation performance

Figure 2 shows the extrapolation errors for the *SPKR*, and other baselines for both classification and regression learning curves. The top row shows the averaged squared error, the bottom row shows the standard deviations. The left column shows the results for the classification learning curves, while the right column the regression results.

SPKR exhibits lower average squared errors made for the targeted extrapolation point compared to baselines across all the Q values that is considered in this work. Similarly, the spread of *SPKR* errors is smaller, except $Q = 10$, where *MMF4* and *BNSL0* and *BNSL1* show lower spread. The increased spread for small training points can be caused by the lack of information selecting the relevant part of the database with little number of training points. Additionally, in our database, highly non-monotonic curves exist where the beginning of the curve is not representative of the end portion. We also observe a slight standard deviation increase for regression problems for our method for when the training set size is increased from 10 to 25, although it still has the lowest spread.

Another observation in Fig. 2 is that *BNSL* performance does not strictly improve as the number of infliction points are increased, hindering the trivial usage of this method in learning curve extrapolation. The varying performances of parametric models for classification and regression learning curves across varying initial segment lengths suggests that our method is able to adjust its bias dynamically with the selected curves.

Since averaging squared errors is prone to being affected by the outliers, we also examine the average rankings in Table 1. Similar to average performance, *SPKR* achieves a better average rank for all initial segment sizes. We observe that our method is highly effective for regression problems with smaller initial segments (Q). Additionally, we investigate the average rank on subsets of the database and found that our model has a better average rank in almost all of the subsets we considered. (See Tables 2 and 3.)

Figure 3 illustrates the cumulative error distribution, where *SPKR* has consistently lower median for all experiments. The only instances where baselines *MMF4* and *WBL4* have lower errors is in the first quartile of $Q = 10$ for classification problems. Finally, we observe that the infliction point increase does not influence the performance of *BNSL* significantly, especially for lower Q values.

Tables 2 and 3 show the average rankings for various subsets of the data. Our method performs well across all the subsets besides the Gaussian Dataset where *MMF4* and *WBL4* has better average ranks. Only, on the Gaussian dataset (*GAU*) our method comes third.

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

Table 1 Average extrapolation ranking (lower is better) for both classification and regression learning curves

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

Table 2 Average rankings (lower is better) of several groupings for the classification learning curves

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

Table 3 Average rankings (lower is better) of various groupings for the regression learning curves

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

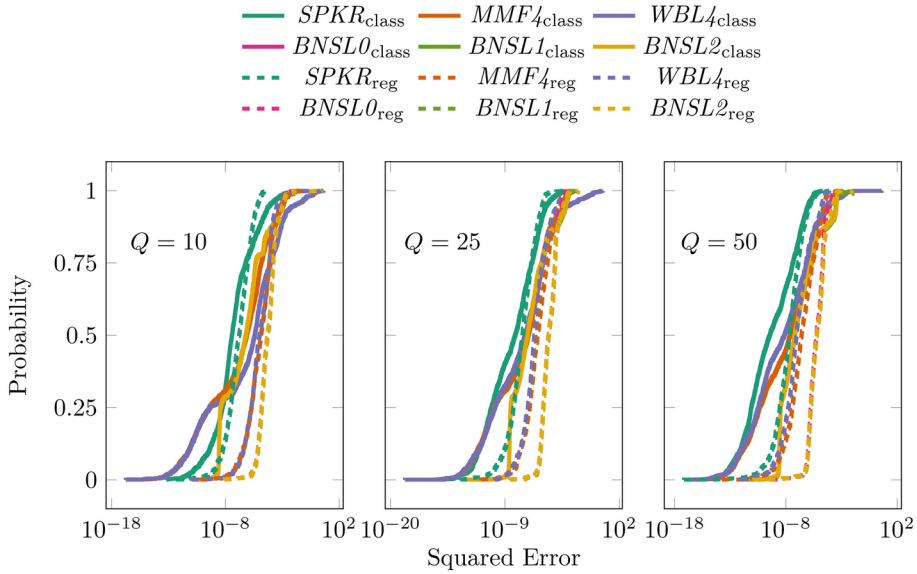


Fig. 3 Empirical cumulative distribution of the extrapolation errors. Solid lines and dashed lines represent the experiments on classification and regression subsets respectively

Table 4 Average ranking for the whole curves (lower is better) for both classification and regression problems

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

respect to whole curve fitting (including all extrapolation and interpolation) based on MSE can be seen in Table 4.

5.2 Obtaining ψ from database

In [28] it is argued that cosine similarity can be a problematic choice in some cases. This is why we investigated two

Table 5 Average extrapolation ranking (lower is better) for the case when we choose the least similar curves in the database

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

Table 6 Average ranking for extrapolation at $N = 100$ (lower is better) for both classification and regression problems

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

Table 7 Average ranking for the whole curves (lower is better) for both classification and regression problems

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

other types of similarity measures. We found that minimizing the area between the targeted curve and the curves in the database, and dynamic time warping [29] does not yield vastly different eigenfunctions ψ in our case. Nonetheless, care must be taken when determining ψ as it the main driving force of the extrapolation performance of the *SPKR*. To demonstrate this, we intentionally choose the most dissimilar curves in our method and observe the average rank of our method drops significantly as shown in 5. Finally, we also attempted to get rid of the similarity measure and extract ψ via *FPCA* on the whole database, we see a similar drop in performance again for the *SPKR*.

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

5.3 Divergence of parametric models

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

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

5.4 Comparison with *MDS*

Performance across classification and regression tasks are presented in Tables 6 and 7. For both types of learning curves, *SPKR* achieved the lowest rankings, outperforming *MDS* in all experiments, except for the classification learning curves with smaller observed part $Q = 10$.

The better performance of our method is expected since it incorporates the *MDS* method. Prediction of *MDS* is based solely on the average of the obtained learning curves from the database. We assumed a solution in the form $\tilde{f} = f + h + \mu_{\mathcal{C}}$, where $\mu_{\mathcal{C}}$ is the mean of the learning curves obtained from the database. Hence, on top of the mean of the most similar curves, we also leverage data points and *FPCA* components from the database to improve our prediction compared to *MDS*. In addition, our method relies on interpolation of the learning curves in the database as it is required for the *FPCA*. This makes our method more robust for cases where the learning curve database might have missing values or follows different sampling strategies

for the sample size N. However, this remains an open question, as our learning curve database and experimental design do not explicitly address such case.

6 Conclusions

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

Although we show that SPKR outperforms the alternative approaches considered in the paper, it just provides a point estimate. In order to get an idea about the uncertainty of the estimate, its probabilistic counterpart, Gaussian Processes can be used (similar to [6]). This would be particularly useful for applications

concerning learning curves such as hyperparameter optimization and model selection. Finally, we present our results using a densely sampled learning curve database without missing values. The effectiveness of our method in cases involving partially missing learning curves remains an open question. As a next step, investigating learning curves with varying sizes or partial observations presents an interesting research direction.

Appendix A: Learning curve database details

In this work, both classification and regression problems are used to create a learning curve database. We introduced variety into the learning curves by altering dataset parameters and model hyperparameters. We obtain our learning curves by using 20 different hyper-parameter and 20 different dataset realizations by regular sampling from the given ranges summarized in Tables 8 and 9. Figures 4 and 5 presents one realization for some of the datasets used. The combinations of models and the datasets used for our learning curve database can be seen in Table 10.

Table 8 Datasets used for the learning curve database

Name	Description
Classification	
Gaussian (GAU)	Artificial 2-class classification problem where both classes are observed from unit multivariate normal $\mathcal{N}(0, \mathrm{I})$. Means of the classes are separated from each other with $p \in [0.1, 5]$
Banana (BAN)	Artificial 2-class classification problem where both classes are observed from mirrored banana shapes. Centers of the two banana shapes are separated from each other with $p \in [0.1, 5]$
Dipping (RDIP-DDIP)	Artificial 2-class classification problem where first class is observed from the unit multivariate normal $\mathcal{N}(0, \mathrm{I})$ and the other class is obtained from a hyper-sphere around the first class with some additional Gaussian noise [9]. This curve is parametrized by the dimensionality $D \in [2, 20]$ and radius of the outer hyper-spherical class $r \in [1, 100]$. See Fig. 4
Regression	
Linear (ELN)	$y = x + \epsilon$ where $x \sim \mathcal{N}(0, 1)$ and $\epsilon \sim \mathcal{N}(0, r)$ with $r \in [0, 1]$
Sine (ESN)	$y = \sin(x) + \epsilon$ where $x \sim \mathcal{N}(0, 1)$ and $\epsilon \sim \mathcal{N}(0, r)$ with $r \in [0, 1]$
Sinc (ESC)	$y = \sin(x)/x + \epsilon$ where $x \sim \mathcal{N}(0, 1)$ and $\epsilon \sim \mathcal{N}(0, r)$ with $r \in [0, 1]$
Sawing (SAW)	Point masses at $(x_a, y_a) = (1, 1)$ and $(x_b, y_b) = (0.1, 1)$ with probabilities $p_a = 0.001$ and $p_b = 1 - p_a$ respectively. See Fig. 5
Gaussian process (DGP)	Special dataset obtained from the prior of a 20 dimensional Gaussian [22]

Table 9 Models and hyperparameters, that are used for the learning curve database

Name	Hyperparameters
Classification	
Nearest mean (NMC)	No hyperparameters
Nearest neighbor (NNC)	Number of neighbours: $\lambda_{\mathrm{nn}} \in [1, 20]$
Linear discriminant (LDC)	Regularization parameter for the covariance matrix: $\lambda_{\mathrm{ld}} \in [10^{-5}, 1]$
Quadratic discriminant (QDC)	Regularization parameter for the covariance matrix: $\lambda_{\mathrm{qd}} \in [10^{-5}, 1]$
Regression	
Linear ridge (LR)	Regularization parameter: $\lambda_{\mathrm{r}} \in [10^{-5}, 1]$
Kernel ridge (KR)	Regularization parameter: $\lambda_{\mathrm{kr}} \in [10^{-5}, 1]$ Gaussian and Laplace kernels with default length-scale $\gamma = 1$
Gaussian process (GP)	Regularization parameter: $\lambda_{\mathrm{kr}} \in [10^{-5}, 1]$ Gaussian kernel with default length-scale $\gamma = 1$
Artificial Neural Network (ANN)	Adam [32] optimizer with learning rate 0.001 and width of the 2 hidden layer network changing between $[5, 25]$

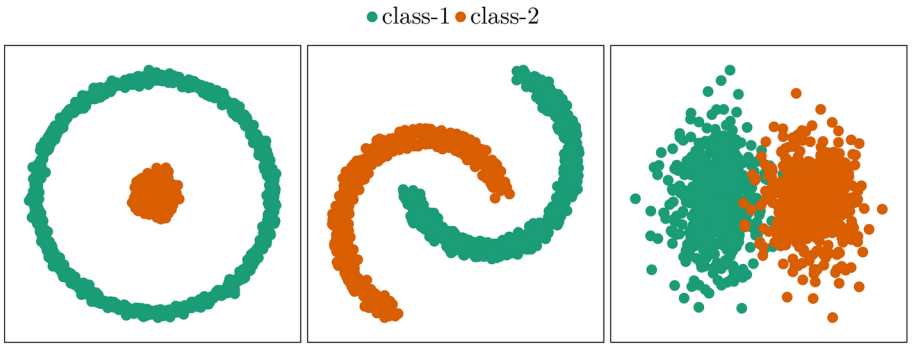


Fig. 4 Scatter plots for one realization of classification datasets (dipping, Gaussian and banana datasets from left to right)

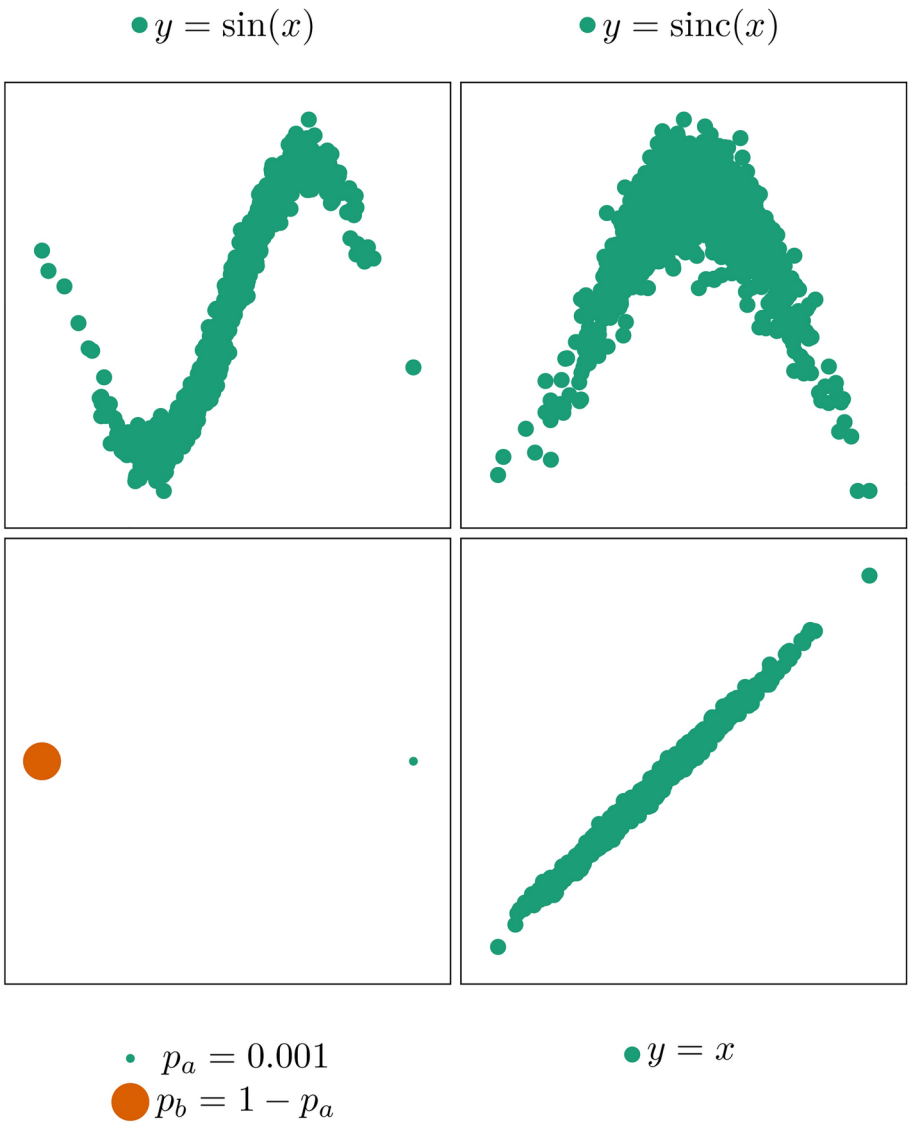


Fig. 5 Scatter plots for one realization of regression datasets (sine, sinc, sawing and linear datasets from left to right)

Table 10 Combinations of datasets and models to create learning curve database

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

Appendix B: Computational details

All of the learning curve generation and majority of experimentation is done in a home-brewed C++ machine learning library that can be found in https://github.com/taylanot/mlcxx.git. If a model that we mention was not available in *mlpack* [31] we code it from scratch (*e.g.* , *NMC*, *LDC*, *QDC*, *KR*, *DGP*). All the datasets used for the database are created by us. *BNSL* [5] and *MDS* [7] are implemented in a library designed specifically for fitting learning curves in the python environment and is accessible at https://github.com/taylanot/learningcurvefitting.git. The learning curve database that we created and the experimental results can be downloaded from https://surfdrive.surf.nl/files/index.php/s/6K4FiCtxeEdvQdx.

Funding No funding was received for conducting this study.

Declarations

Conflict of interest The authors declare that they have no competing financial or non-financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Mohr F, Viering TJ, Loog M, Rijn JN (2023) Lcdb 1.0: an extensive learning curves database forÂ classification tasks. In: Amini M-R, Canu S, Fischer A, Guns T, Kralj Novak P, Tsoumakas G (eds) Machine learning and knowledge discovery in databases, pp 3–19. Springer, Cham
2. Viering TJ, Loog M (2021) The shape of learning curves: a review. CoRR. arxiv:abs/2103.10948
3. Loog M, Viering T (2022) A survey of learning curves with bad behavior: or how more data need not lead to better performance. https://arxiv.org/abs/2211.14061
4. Adriaensen S, Rakotoarison H, Müller S, Hutter F (2023) Efficient Bayesian learning curve extrapolation using prior-data fitted networks. https://arxiv.org/abs/2310.20447
5. Caballero E, Gupta K, Rish I, Krueger D (2023) Broken neural scaling laws. https://arxiv.org/abs/2210.14891
6. Harvey E, Chen W, Kent DM, Hughes MC (2023) A probabilistic method to predict classifier accuracy on larger datasets given small pilot data. https://arxiv.org/abs/2311.18025
7. Leite R, Brazdil P (2005) Predicting relative performance of classifiers from samples. In: Proceedings of the 22nd International Conference on Machine Learning - ICML '05, pp 497–503. ACM Press, Bonn, Germany. https://doi.org/10.1145/1102351.1102414 . Accessed 2024-11-25
8. Chen Z, Loog M, Krijthe JH (2023) Explaining two strange learning curves. In: Calders T, Vens C, Lijffijt J, Goethals B (eds) Artificial intelligence and machine learning. Springer, Cham, pp 16–30
9. Loog M, Duin RPW (2012) The dipping phenomenon. In: Gimel'farb G, Hancock E, Imiya A, Kuijper A, Kudo M, Omachi S, Windeatt T, Yamada K (eds) Structural, syntactic, and statistical pattern recognition. Springer, Berlin, pp 310–317
10. Ruhkopf T, Mohan A, Deng D, Tornede A, Hutter F, Lindauer MT (2023) Masif: meta-learned algorithm selection using implicit fidelity information. Trans Mach Learn Res
11. Jawed S, Jomaa H, Schmidt-Thieme L, Grabocka J (2021) Multi-task learning curve forecasting across hyperparameter configurations and datasets. In: Oliver N, Pérez-Cruz F, Kramer S, Read J, Lozano JA (eds.) Machine Learning and Knowledge Discovery in Databases. Research Track, pp 485–501. Springer, Cham
12. Domhan T, Springenberg JT, Hutter F (2015) Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: International Joint Conference on Artificial Intelligence. https://api.semanticscholar.org/CorpusID:369457
13. Klein A, Falkner S, Springenberg JT, Hutter F (2017) Learning curve prediction with bayesian neural networks. In: International Conference on Learning Representations. https://openreview.net/forum?id=S11KBYclx
14. Egele R, Guyon I, Sun Y, Balaprakash P (2023) Is one epoch all you need for multi-fidelity hyperparameter optimization?. https://arxiv.org/abs/2307.15422
15. Yan S, White C, Savani Y, Hutter F (2021) NAS-Bench-x11 and the power of learning curves. https://doi.org/10.48550/arXiv.2111.03602arXiv. arXiv:2111.03602 [cs]. Accessed 2024-11-25
16. Lee DB, Zhang AS, Kim B, Park J, Lee J, Hwang SJ, Lee HB (2024) Cost-sensitive multi-fidelity bayesian optimization with transfer of learning curve extrapolation. arXiv. arXiv:2405.17918 [cs]. https://doi.org/10.48550/arXiv.2405.17918. Accessed 2024-11-25
17. Perlich C (2010) In: Sammut C, Webb GI (eds) Learning curves in machine learning, pp 577–580. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_452
18. Jain A, Swaminathan G, Favaro P, Yang H, Ravichandran A, Harutyunyan H, Achille A, Dabeer O, Schiele B, Swaminathan A, Soatto S (2023) A meta-learning approach to predicting performance and data requirements. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 3623–3632. https://doi.org/10.1109/CVPR52729.2023.00353

19. Schölkopf B, Smola AJ (2001) Learning with Kernels: support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge
20. Pearson K (1901) Liii. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2(11):559–572. https://doi.org/10.1080/14786440109462720
21. Billheimer D (2007) Functional data analysis, 2nd Edition Edited by J. O. Ramsay and B. W. Silverman. Biometrics 63(1):300–301. https://doi.org/10.1111/j.1541-0420.2007.00743_1.xhttps://academic.oup.com/biometrics/article-pdf/63/1/300/52300836/biometrics_63_1_300.pdf
22. Sollich P (2001) Gaussian process regression with mismatched models. https://arxiv.org/abs/cond-mat/0106475
23. Gu B, Hu F, Liu H (2001) Modelling classification performance for large data sets. In: Wang XS, Yu G, Lu H (eds) Advances in web-age information management. Springer, Berlin, pp 317–328
24. Kolachina P, Cancedda N, Dymetman M, Venkatapathy S (2012) Prediction of learning curves in machine translation. In: Li H, Lin C-Y, Osborne M, Lee GG, Park JC (eds.) Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 22–30. Association for Computational Linguistics, Jeju Island, Korea. https://aclanthology.org/P12-1003
25. Liu DC, Nocedal J (1989) On the limited memory bfgs method for large scale optimization. Math Program 45:503–528
26. Kielhöfer L, Mohr F, Rijn JN (2024) Learning curve extrapolation methods across extrapolation settings. In: Miliou I, Piatkowski N, Papapetrou P (eds) Advances in intelligent data analysis XXII. Springer, Cham, pp 145–157
27. Nadaraya EA (1964) On estimating regression. Theory Prob Appl 9(1):141–142. https://doi.org/10.1137/1109020
28. Steck H, Ekanadham C, Kallus N (2024) Is cosine-similarity of embeddings really about similarity? In: Companion Proceedings of the ACM Web Conference 2024. WWW '24, pp. 887–890. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3589335.3651526
29. Bringmann K, Fischer N, Hoog I, Kipouridis E, Kociumaka T, Rotenberg E (2023) Dynamic dynamic time warping. https://arxiv.org/abs/2310.18128
30. Leite R, Brazdil P (2010) Active testing strategy to predict the best classification algorithm via sampling and metalearning. In: Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence, pp 309–314. IOS Press, NLD
31. Curtin RR, Edel M, Shrit O, Agrawal S, Basak S, Balamuta JJ, Birmingham R, Dutt K, Eddelbuettel D, Garg R, Jaiswal S, Kaushik A, Kim S, Mukherjee A, Sai NG, Sharma N, Parihar YS, Swain R, Sanderson C (2023) mlpack 4: a fast, header-only c++ machine learning library. J Open Source Softw 8(82):5026. https://doi.org/10.21105/joss.05026
32. Kingma DP, Ba J (2017) Adam: a method for stochastic optimization. https://arxiv.org/abs/1412.6980