LCDB 1.0

An Extensive Learning Curves Database for Classification Tasks

Mohr, Felix; Viering, Tom J.; Loog, Marco; van Rijn, Jan N.

DOI
10.1007/978-3-031-26419-1_1

Publication date
2023

Document Version
Final published version

Published in
Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2022, Proceedings

LCDB 1.0: An Extensive Learning Curves Database for Classification Tasks

Felix Mohr[1], Tom J. Viering[2(✉)], Marco Loog[2,3], and Jan N. van Rijn[4]

[1] Universidad de La Sabana, Chía, Colombia
felix.mohr@unisabana.edu.co
[2] Delft University of Technology, Delft, The Netherlands
{t.j.viering,m.loog}@tudelft.nl
[3] University of Copenhagen, Copenhagen, Denmark
[4] Leiden University, Leiden, The Netherlands
j.n.van.rijn@liacs.leidenuniv.nl

Abstract. The use of learning curves for decision making in supervised machine learning is standard practice, yet understanding of their behavior is rather limited. To facilitate a deepening of our knowledge, we introduce the Learning Curve Database (LCDB), which contains empirical learning curves of 20 classification algorithms on 246 datasets. One of the LCDB's unique strength is that it contains all (probabilistic) predictions, which allows for building learning curves of arbitrary metrics. Moreover, it unifies the properties of similar high quality databases in that it (i) defines clean splits between training, validation, and test data, (ii) provides training times, and (iii) provides an API for convenient access (pip install lcdb). We demonstrate the utility of LCDB by analyzing some learning curve phenomena, such as convexity, monotonicity, peaking, and curve shapes. Improving our understanding of these matters is essential for efficient use of learning curves for model selection, speeding up model training, and to determine the value of more training data.

Keywords: Learning curves · AutoML · Meta-learning

1 Introduction

Learning curves provide the prediction performance of a learning algorithm against the dataset size it has used for training. Such curves provide essential information for decision making [27,35]. For example, they can be extrapolated to determine the value of gathering more data or can be used to speed up training by selecting a smaller dataset size that still reaches sufficient accuracy. In addition, learning curves can provide useful information for model selection [2,26]. Particularly important questions concern the performance in the limit and the training set size at which the learning curves of two algorithms cross, as this can

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-26419-1_1.

M.-R. Amini et al. (Eds.): ECML PKDD 2022, LNAI 13717, pp. 3–19, 2023.
https://doi.org/10.1007/978-3-031-26419-1_1

tell us when one learning algorithm should be preferred over the other. To make use of learning curves for these purposes, it is essential that we know their shape (exponential, power law, etc.) so that we can reliably extrapolate or interpolate them for these tasks.

Unfortunately, there is a gap between common assumptions made by methods used for the above purposes and empirical evidence that would justify those assumptions. For instance, a common assumption is that learning curves behave well, i.e., more data leads to better performance. This assumptions is made for example in the extrapolation of learning curves for decision making [8,17,26]. However, a recent study [35] has collected a variety of results from literature, which illustrate that learning curves can display surprising shapes, such as multiple local minima, peaks, or curves that deteriorate with more data. These examples all illustrate that, in practice, there are clear gaps in our understanding of learning curves and their potential behavior. This limits the practical use of such curves, since correct extrapolations or interpolations depend crucially on the accuracy of learning curve models.

Empirical knowledge on learning curves is surprisingly scarce and often afflicted with severe limitations. Two recent surveys [27,35] give an extensive overview of the learning curve literature including empirical studies. One of the main problems of current studies is that the number of datasets and/or algorithms considered is small: Gu et al. [14] study two classifiers and eight datasets, Perlich et al. [30] examine two classifiers and 36 datasets, Li [23] use eight classifiers on three datasets, and recently Brumen et al. [3] considered four classifiers on 130 datasets. Other issues include that (i) the data acquired is not openly shared, hence not easily accessible, (ii) the focus is on specific performance metrics, or (iii) only a single train/test split is considered, providing a weak estimate for the out-of-sample learning curve and limiting the reliability of analysis.

To overcome these limitations and facilitate research on learning curves and their behavior, this work introduces the Learning Curve Database (LCDB), a high quality and extensive database of classification task learning curves. The current version of LCDB provides already over 150 GB of *ground truth and prediction vectors* of 20 classification algorithms from the scikit-learn library on 246 datasets. These prediction vectors have been recorded for models being trained on an amount of instances (called anchors) that are rounded multiples of $\sqrt{2}$. Instead of training each learner only once at each anchor, we created 25 splits in order to be able to obtain more robust insights into the out-of-sample performance. This makes it, in various respects, the biggest database for learning curves available today.

Note that, in the context of neural networks, a database called LCBench (Learning Curve Benchmark) [37] contains the performance of neural nets versus the number of epochs. Following [35] we call such curves *training curves*, to illustrate the difference with our learning curves which plot performance versus training set size. Because the LCDB investigates learning curves, and LCBench investigates training curves, these databases are incompareable.

LCDB is the first to provide (probabilistic) predictions together with the ground-truth labels. The availability of probabilistic predictions makes it possible

to compute one's own choice of metrics, like AUROC or log-loss, rather then have to deal with precomputed ones. Curve data is provided for 25 stratified splits at each anchor for training, validation, and test data, enabling the construction of different curve types. Moreover, runtimes are provided for model training and prediction to study its dependence on sample size.

Other benefits of LCDB are that it enables (i) the a-posteriori analysis of the *shape* of curves for different learners and of function classes describing such shapes, (ii) the study of the relationship between training and test curves, (iii) the simultaneous analysis of learning curves, e.g., whether or not they intersect or if such intersection can be predicted, (iv) research into principled models for the runtime behavior of the algorithms, (v) benchmarking algorithm selection problems, and (vi) quick insights into the "difficulty" of datasets, which can be useful for the design of such benchmarks.

To showcase the utility of LCDB, we provide initial results on three of the above-mentioned analyses: (i) a study of the presence of three shape properties: monotonicity, convexity, and peaking, (ii) the identification of crossing behavior for all pairs of learners on all datasets, and (iii) an analysis of the goodness of fit of a set of model classes, both for the case of capturing the whole curve and predicting higher anchor performance from a set of lower anchor performances. A major insight of the study is that there is great support for the hypothesis that error-rate curves are largely (even though usually not perfectly) monotonic, convex, and mostly free of peaks (double descent). A second major insight is that typical learning curve models considered in literature, such as the 2 or 3 parameter power law, exponential and logarithmic models, may be significantly outperformed by 4 parameter models when used for extrapolation. We find that `mmf4` performs the best overall with `wbl4` a close second, but our results should be interpreted with care, as we ran into various issues with fitting.

2 The Learning Curve Database

To understand and motivate the way how LCDB is designed, Sect. 2.1 briefly recalls the formal definition of the learning curves and terminology. While an intuition on learning curves is common sense, recent surveys [27,35] show that there is a variety of performance curves (with similar terminology), which can quickly lead to confusion on what the exact subject of interest is. Our learning curves plot generalization performance versus training set size.

2.1 Formal Background on Learning Curves

Out-of-Sample Learning Curves. We consider learning curves in the context of supervised machine learning. We follow the definition of Mohr and van Rijn [27], which assumes some *instance space* $\mathcal{X}$ and a *label space* $\mathcal{Y}$. A *dataset* $D \subset \{(x, y) \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$ is a *finite* relation between the instance space and the label space. We denote with $\mathcal{D}$ the set of all possible datasets. A *learning algorithm* is a function $a : \mathcal{D} \times \Omega \to H$, where $H = \{h \mid h : \mathcal{X} \to \mathcal{Y}\}$ is the space of hypotheses and Ω is a source of randomness, such as the random seed.

The performance of a *learning algorithm* is the performance of the hypothesis it produces. For a hypothesis h, this performance is measured as $\mathcal{R}_{out}(h) := \int_{\mathcal{X},\mathcal{Y}} loss(y, h(x))\, d\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$. Here, $loss(y, h(x)) \in \mathbb{R}$ is the penalty for predicting $h(x)$ for instance $x \in \mathcal{X}$ when the true label is $y \in \mathcal{Y}$, and $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$ is the joint probability measure $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$ on $\mathcal{X} \times \mathcal{Y}$ underlying the analyzed data. To assess the performance of a learner a, we average the performance once more over the input data and randomness of the learner, which determine the hypothesis:

$$\mathcal{C}(a, s) = \int \mathcal{R}_{out}(a(D_{tr}, \omega)) d\mathbb{P}^s_{\mathcal{X}\times\mathcal{Y}} d\mathbb{P}_{\Omega}, \tag{1}$$

where $d\mathbb{P}^s_{\mathcal{X}\times\mathcal{Y}}$ is the distribution over i.i.d. sampled training sets D_{tr} of size s.

Considering Eq. (1) as a function of the number of observations for a fixed learner a yields a *learning curve* of learner a. That is, the observation learning curve is the function $\mathcal{C}(a, \cdot) : \mathbb{N} \to \mathbb{R}$, which is a *sequence* of performances, one for each training size. There are other types of closely related curves, most notably those based on the number of *iterations* or *epochs* used during training (training curves), which we do not consider. See [27,35] for details.

Empirical Learning Curves. While we are generally interested in the *true* learning curve defined in Eq. (1), we cannot determine it in practice. First, the out-of-sample error $\mathcal{R}_{out}$ is unknown because the measure $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$ is unknown. Next, the necessity to average over the oftentimes uncountable set of all possible train sets and random seeds adds additional problems.

In practice, we therefore have to rely on *empirical learning curves*. An empirical learning curve is any set of estimates of a true learning curve at different training set sizes, which are called *anchors*. At anchor s, this estimate is obtained by (i) creating one (hold-out) or several (cross-validation) splits (D_{tr}, D_{te}) such that $|D_{tr}| = s$, (ii) obtaining for each such split a hypothesis via $h = a(D_{tr}, \omega)$ using a unique random seed each time, and (iii) then computing the *empirical risk* $\mathcal{R}_{in}(h, \tilde{D}) := \frac{1}{|\tilde{D}|} \sum_{(x,y)\in\tilde{D}} loss(y, h(x))$. Averaging these estimates for $\tilde{D} = D_{te}$ yields an estimate of the curve in Eq. (1) at anchor s:

$$\widehat{\mathcal{C}}(s) := \widehat{\mathcal{C}}(a, D, s) = \frac{1}{k} \sum_{i=1}^{k} \mathcal{R}_{in}(a(D^i_{tr}, \omega_i), D^i_{te}). \tag{2}$$

Since it is usually clear from the context which learner a and dataset D are used, we typically omit them in the notation and only write $\widehat{\mathcal{C}}(s)$. Note that we can also use $\tilde{D} = D_{tr}$ to approximate a train curve, or even use a mixture of both, instead of the test performance curve.

LCDB stores, for all points in D, the ground truth and prediction obtained from a trained learner (hypothesis) $a(D^i_{tr}, \omega_i)$ for various splits. With this, the empirical learning curve in Eq. (2) can be recovered for any concrete measure $\mathcal{R}_{in}$, and replacing D^i_{te} with D^i_{tr} also allows us to compute the curve of training performances. Thereby and in contrast to previous works relying only on one split [3], LCDB can be used to approximate the out-of-sample curve more reliably.

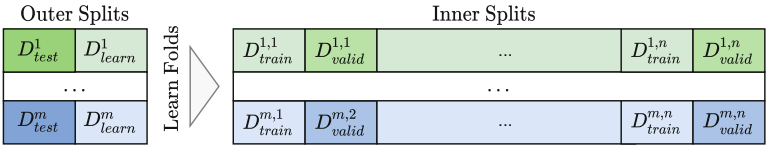

Fig. 1. Procedure to create training data for learners in LCDB: First an 90%/10% outer split is sampled. The 90% learning data fold is further sampled into different independent 90%/10% splits for essential training and validation respectively. Training data at different anchors is sampled independently from $D^{i,j}_{train}$.

2.2 Data Collection Methodology

The goal of LCDB is to provide learning curves for *model selection processes*. In model selection, it is usually not enough to have test data, but at time of selecting a model, one typically evaluates models, which requires *validation data*. If the model selection technique itself is supposed to be cross-validated, one arrives at a concept called *nested cross validation*. To simulate such processes, we need an explicit separation between training, validation and test data.

With this in mind, LCDB does *not* increase the data available for training in a monotonic way, as for example employed in [3], but assumes *independent* training portions at every anchor. That is, when training a classifier on the set S_1 of instances and later on the set S_2 of instances, where $|S_1| < |S_2|$, then we do not want that $S_2 \supset S_1$. Instead, we conceive that the training samples used for S_1 and S_2 are drawn *independently* from a pool of available training instances. While a monotonic approach makes sense in the context of data acquisition, for model selection we are interested in estimates of the out-of-sample performance, which is not compatible with systematic dependencies of the training data used at different points on the learning curve.

To reach this goal and also incorporate the idea of validation data, LCDB is based on nested splits itself. The procedure is illustrated in Fig. 1. On each dataset, we consider m *outer* splits of 90%/10%. These splits help to assess the performance of, say, model selection approaches in a simulated cross-validation. Note that all splits in LCDB are stratified. The test folds do not need to be disjoint and we simply create several independent hold-out splits. Since we need to store all the predictions, the test fold is essentially bounded to 5000 instances in LCDB, which we consider to be large enough for a satisfying approximation. Now let $(D^i_{learn}, D^i_{test})$ the i-th such split. From the learning set D^i_{learn}, n further *inner* splits of 90%/10% are derived. The inner splits can serve, for instance, to simulate cross-validations conducted by the model selection approach itself. Let $D^{i,j} = (D^{i,j}_{train}, D^{i,j}_{valid})$ be the j-th such inner split derived from the i-th outer split. Note that [3] neither compute outer nor several inner splits but basically create *one* 80%/20% split.

For each training set, we compute the prediction vectors and probabilities according to a geometric schedule [31] for all three sets: training, validation, and test. Formally, for each train set $D^{i,j}_{train}$ with $1 \leq i \leq m, 1 \leq j \leq n$, we create

a series $D^{i,j,1}_{train}, D^{i,j,2}_{train}, \ldots$ so that $s_k := |D^{i,j,k}_{train}| = \lceil 2^{\frac{7+k}{2}} \rceil$, where the smallest dataset size is 16 (for $k = 1$). A learner is trained using $D^{i,j,k}_{train}$, and then the prediction vectors (and probabilities) are computed for all instances in $D^{i,j,k}_{train}$, $D^{i,j}_{valid}$, and D^{i}_{test}, respectively. From these, it is possible to compute arbitrary metrics on the training data, the validation data, and the test data afterwards. In the initial setup of LCDB, we have $m = n = 5$; further splits can be reliably added in the future thanks to the seeded architecture of the LCDB code.

Note that, at the time of working with LCDB, the distinction between outer and inner splits is optional and can be omitted if one simply wants to get empirical learning curves for learners. To obtain the performance at a particular anchor, one can proceed as follows. For each of the $m \times n$ train sets $D^{i,j,k}_{train}$ for anchor s_k, the prediction vectors of the validation set $D^{i,j}_{valid}$ and the test set D^{i}_{test} after having trained on $D^{i,j,k}_{train}$ are available. These can simply be merged in order to compute some metric, say, the error rate at s_k. This gives $m \cdot n$ performance estimates for each anchor.

2.3 Datasets

In this initial version, LCDB contains learning curves for 246 datasets. The large majority of these datasets was already used in AutoML [9] or is part of published benchmarks [13]. Our main criterion for the source of the data is API-based reproducibility, in order to enable a closed algorithm that is able to fully reproduce results without the need of manually downloading the datasets. To this end, we chose OpenML.org [34] as a source, which, in contrast to other repositories like UCI, offers an official API in both Java and Python. The datasets themselves represent a large range over various properties such as the number of instances, features, and classes. For details, we refer to the supplement.

Many datasets need to be preprocessed before learning can take place. To cope with that, we applied two (admittedly arbitrary) pre-processing steps. First, all missing values were replaced by the column median for numerical and by the column mode for categorical attributes. Then, all categorical attributes were binarized into a Bernoulli encoding. Features were not scaled, as we have not implemented pipelines yet. Classifiers sensitive to scaling of features (such as Nearest Neighbours, SVM's, etc.) may be disadvantaged.

2.4 Classifiers

In this initial work, we considered only classifiers from the scikit-learn library [29]. Using their default hyper-parameters, the 20 considered classifiers are (we here show class names and, in parentheses, abbreviations used in figures and the discussion): SVC (linear, poly, rbf, sigmoid), LinearDiscriminantAnalysis (LDA), QuadraticDiscriminantAnalysis (QDA), ExtraTreesClassifier (Extra Trees), GradientBoostingClassifier (Grad. Boost), RandomForestClassifier (RF), LogisticRegression (LR), PassiveAggressiveClassifier (PA), Perceptron, RidgeClassifier

(Ridge), SGDClassifier (SGD), BernoulliNB, MultinomialNB, KNeighborsClassifier (kNN), MLPClassifier (MLP), DecisionTreeClassifier (DT), ExtraTreeClassifier (Extra Tree). Note the difference between ExtraTreeClassifier (single tree) and ExtraTreesClassifier (ensemble).

2.5 Availability via PyPI (pip)

The learning curves are available[1] in two formats. First, there are the raw sets of ground truth and prediction vectors, which can be downloaded in separate zip files, one for each dataset. These files are space-consuming as for some datasets, the learning curve files accumulate to more than 10GB when unpacked. The second option is to download learning curves that have been pre-compiled for specific metrics such as accuracy, log-loss, and the F1-measure. In the latter format, the data can also be accessed directly through an API in Python via a package that can be installed with `pip install lcdb`.

3 Illustrative Usage of LCDB

To efficiently perform model selection, or to estimate the added value of more data collection, etc., it is essential to develop good and accurate insight into learning curve behaviour. We consider the following three, relevant questions.

1. What is the probability, for each learner, to exhibit a learning curve that is (i) monotone, (ii) convex, or (iii) peaked?
2. What is the probability that learner A starts off worse than learner B, but exhibits better performance on the full data, i.e., the learning curve of A crosses the one of B (from above)?
3. What is the goodness of fit/extrapolation performance of some learning curve models when interpolating or extrapolating the curve?

We focus on error-rate curves, but LCDB can easily facilitate other metrics. Our interest is in answering the above questions for the true, i.e., out-of-sample learning curves, as defined in Eq. (1), and LCDB allows us to estimate these by averaged empirical curves $\widehat{\mathcal{C}}$ as per Eq. (2). The mean is here formed across both the different outer and inner splits of the data. Since the initial version of LCDB comes with $5 \cdot 5 = 25$ training sets, we get a much better estimate of the mean compared to previous works, such as [3] that only consider a single split. In what follows, we formulate the necessary definitions immediately in terms of $\widehat{\mathcal{C}}$ as a proxy of $\mathcal{C}$.

3.1 Curve Properties: Monotonicity, Convexity, and Peaking

Background. Knowledge about properties like monotonicity [24], convexity [26], and peaking is important to justify particular extrapolation techniques

[1] Raw data: https://openml.github.io/lcdb/; Code: https://github.com/fmohr/lcdb.

based on optimistic extrapolation [26] or curve models that assume such shapes, like the inverse power law, exponential, or logarithmic functions [8,17]. We here consider decreasing learning curve models as typically expected for error rates, but the modification of our experiments to alternative models is straightforward.

While all three properties are binary and either are or are not present for a concrete function, it is helpful to consider some *degree* of monotonicity and of convexity. For instance, if the learning curve only has some tiny oscillation somewhere in the curve but is monotone and convex anywhere else, we would still want to consider it largely well-behaved, even though not perfect.

Clearly, there is no unique way in which a degree of monotonicity or convexity can be measured and we merely propose some specific operationalizations. Whether or not these are meaningful eventually depends on the context of application. Now, assuming descending curves (an assumption implicit in its original definition), we take as the degree of violation of monotonicity the highest positive increase observed at any anchor on the empirical curve:

$$\epsilon_{mono} = \max_{s_i, i<T} \left\{ \max \left(0, \widehat{\mathcal{C}}(s_{i+1}) - \widehat{\mathcal{C}}(s_i) \right) \right\},$$

where T is the highest available anchor index. Similarly, we compute such a violation for convexity:

$$\epsilon_{conv} = \max_{s_i, i<T-1} \left\{ \max \left(0, \widehat{\mathcal{C}}(s_{i+1}) - \tfrac{1}{2} \left(\widehat{\mathcal{C}}(s_i) + \widehat{\mathcal{C}}(s_{i+2}) \right) \right) \right\}.$$

We note that, for learning curves over a finite range of anchors, monotonicity and convexity are mutually neither necessary nor sufficient criteria. To illustrate this, the bottom right plot in Fig. 2 shows two examples of learning curves that are monotone but not convex and two examples that are convex but not monotonically decreasing.

Peaking [35], also called sample-wise double descent/ascent, is a classic phenomenon, currently studied in neural networks [25,28]. It refers to the curve showing worsened performance, after initial performance improvements, before eventually starting to consistently improve again at larger anchor sizes. It has been observed for different types of performance curves, including learning curves as functions of the sample size used for training [28]. In a sense, peaking is a special form of non-monotonicity that often occurs at a specific location in the learning curve, typically around the point where model complexity and sample size coincide. Given its prominence, we study it also separately. While one could, in principle, measure the extent of temporary deterioration, we here choose to measure whether such a phenomenon occurs or not in a binary fashion:

$$peak = \left[\!\left[\exists i < T : \widehat{\mathcal{C}}(s_i) < \widehat{\mathcal{C}}(s_{i+1})) \wedge \left(\exists u > i, \forall v \geq u : \widehat{\mathcal{C}}(s_{v-1}) > \widehat{\mathcal{C}}(s_v)) \right) \right]\!\right].$$

Results. The top row of Fig. 2 summarizes the monotonicty (left) and convexity (right) results over all learners and datasets, effectively giving the general cumulative distribution of the maximum violation. The different colors show

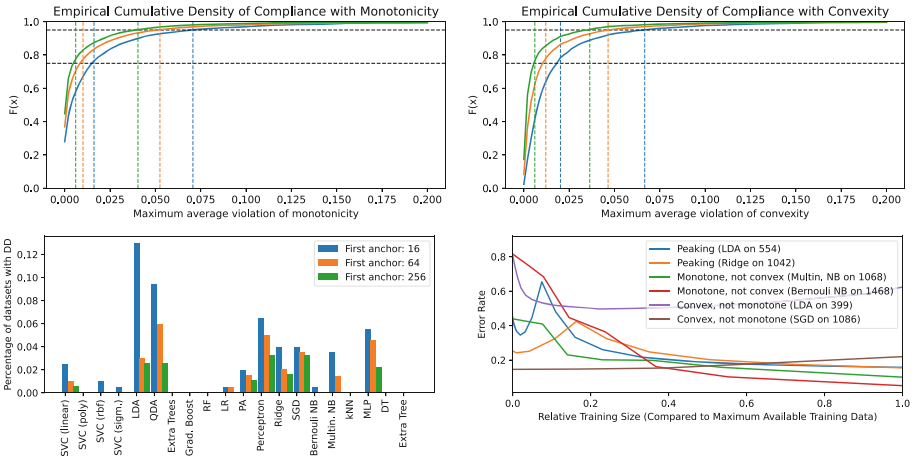

Fig. 2. Insights into shape properties of learning curves. Numbers in legend are dataset IDs at OpenML.org. Top row: cumulative distribution of the degree of violation of monotonicity (left) and concavity (right) for different entry points of analysis (blue, orange green for analysis starting at anchor 16, 64, and 256, respectively). Bottom left: Relative frequency (across datasets) of peaking per learner. Bottom right: Some sample learning curves with non-standard properties. (Color figure online)

different entry points of analysis. Clearly, if we were to consider learning curves starting from a single training instance, we would expect high variation in the observations and possibly non-monotonicities even in the true curve. For this reason, it can be meaningful to start to study the learning curve only past some minimum training size [20,21,32]. As can be seen in the figure, the probability of violations is significantly reduced with increasing entry point from 16 (blue) to 256 (green) via 64 (orange).

The bottom left panel of Fig. 2 provides insights into peaking. It shows, per learner, how often peaking can be observed, i.e., the relative number of datasets on which it is observed. As can be seen, sample-wise peaking in error-rate curves is rather rare and many learners do not exhibit this property at all. The right plot shows some example of curves where peaking was detected. It also shows two examples failing to be monotone and two failing to be convex.

3.2 Learning Curve Crossing

We now consider the probability that some learner A will outperform learner B at the highest available training size (effectively $|D^{i,j}_{train}|$) given that it has a worse initial performance.

The results are summarized in Fig. 3. It shows that there is indeed quite a number of algorithms that start off worse than others, but recover and exceed the performance of the competitor. In particular, for learners commonly considered strong, such as Gradient Boosting or Random Forests, there is a high probability

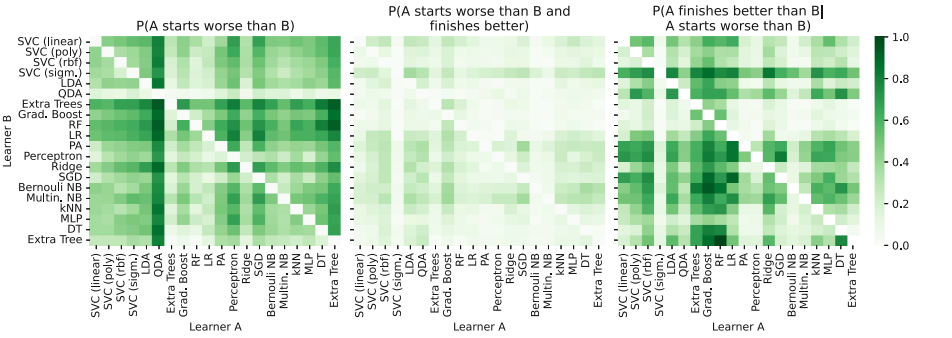

Fig. 3. Probabilities associated with the two events (i) that learner A (columns) starts off worse than learner B (rows) and (ii) that learner A finishes better than learner B. For each combination of learners for A and B, the left plot shows the empirical probability of event (i), the middle plot the joint probability of events (i) and (ii), and the right plot the probability of (ii) given (i). Darker colors show higher probabilities. (Color figure online)

to eventually outperform the other learners (dark green columns). The opposite is hardly ever the case (light green rows). For other algorithms, the opposite is the case, e.g., the Perceptron hardly ever takes the lead if it starts off bad (light green column), but is often outperformed in the long run, even if starts off comparatively well (dark green row).

The above formulation focuses on the value of the learning curve at the maximum training size available, but it could be the case that one or even both of the learning curves have not yet converged at that training size. In another view, one could ask for extrapolation models and then answer the same question as above, based on some potential training size that exceeds the available data.

3.3 Learning Curve Fitting and Extrapolation

Background. Many parametric models have been proposed in literature for modeling learning curves [35]. Typical examples are power laws, exponentials and logarithmic models. Table 1 provides an overview, where `last1` is a baseline that takes the last point on the learning curve that it observes, and always predicts this value. This is a simple baseline that is also used in [22] for example.

Several benchmarks have been performed to determine the best parametric model for empirical learning curves [3,5,10,14,18,19,33], but all have their limitations [35]. First, most studies benchmark only 3 to 6 curve models [3], or leave out models with bias terms, so that non-zero asymptotic cannot be modeled [10]. A comprehensive experiments encompassing all models proposed remains missing, making any conclusive comparison difficult. Additional complications arise from the fact that some studies [6,10,19,33] fit exponentials or power laws by transforming the data and performing the usual least squares procedure in the transformed space [35]. Finally, not all works evaluate the results of curve

Table 1. The 16 learning curve models under consideration. x is the size of the training set and a, b, c, d are parameters to be estimated. Some model performance increase rather than decrease. last1 is a baseline (horizontal line), the number in the abbreviation indicates the number of parameters.

Model	Formula	Used in	Model	Formula	Used in
`last1`	a	[22]	`vap3`	$\exp\left(a + \frac{b}{x} + c\log(x)\right)$	[14]
`pow2`	$-ax^{-b}$	[10,14,19,33]	`expp3`	$c - \exp\left((-b+x)^a\right)$	[18]
`log2`	$-a\log(x) + b$	[3,4,10,14,19,33]	`expd3`	$c - (-a+c)\,exp(-bx)$	[18]
`exp2`	$a\exp(-bx)$	[10,19,33]	`logpow3`	$a/\left((x\exp(-b))^c + 1\right)$	[8]
`lin2`	$ax + b$	[4,10,19,33]	`pow4`	$a - b\,(d+x)^{-c}$	[18]
`ilog2`	$-a/\log(x) + b$	[18]	`mmf4`	$(ab + cx^d)/(b + x^d)$	[14]
`pow3`	$a - bx^{-c}$	[14,18]	`wbl4`	$-b\exp(-ax^d) + c$	[14]
`exp3`	$a\exp(-bx) + c$	[3,4,18]	`exp4`	$c - \exp(-ax^d + b)$	[18]

fitting with help of statistical tests [14,18,19], which casts some doubt on the significance of their findings. Most studies found evidence that learning curves are modeled best by `pow2`, `pow3`, `log2` or `exp3` [35].

In this work, we therefore aim to compare all parametric models for learning curves so far suggested in the literature, perform model evaluation on unseen curve data (to avoid overfitting concerns [35]), and we aim to use a careful fitting approach, which is the same for all models (except exp4 - see Supplement), and perform statistical tests to determine significance of our findings.

We evaluate curve fitting in terms of the Mean Squared Error (MSE) with (1) the points seen during curve fitting (train anchors, so interpolation), (2) the points that are not seen during fitting (test anchors, extrapolation), (3) extrapolating to the last anchor. This is a common procedure [3,14] and we use averaged curves (25 splits). One thing to note is that curve fitting is more difficult than one may expect, to cope with this we use 5 random restarts and we discard failed fits (MSE>100). In the following, we focus on the main findings; an extensive report with many more details can be found in the Supplement.

Results. Our main findings are summarized in Figs. 4 and 5 and Table 2. Since the semantics differ between the figures, we discuss them one at a time.

Figure 4 shows the performance for the different curve models in the form of (empirical) cumulative densities with respect to the prediction error. Here, we aggregate over models that were fit with all available anchors up to a size of 20% of the maximum available training size for fitting. Both plots show the curves that are at some point of time on top of the others in solid and all the others (somewhat dominated models) in dotted curves. The left plot shows how well the models can accommodate to the anchors they have seen (the training MSE). Unsurprisingly, we find that the more parameters the better the fits, except for `exp3` who performs as well as the 2 parameter models.

The right plot shows how well each model can predict the performance for all upcoming anchors (on average). Here, roles have changed somewhat. It indicates

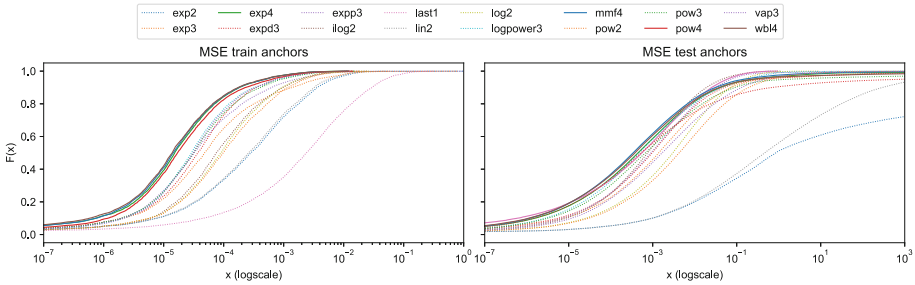

Fig. 4. The Cumulative Density Functions (CDF) for the Mean Squared Error (MSE) for all curve models for interpolation on the train anchors and extrapolation to all test anchors, summarized over all curve fitting experiments.

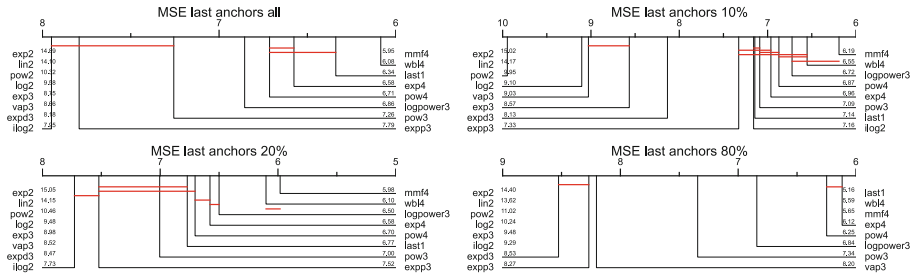

Fig. 5. Critical diagrams for the ranks for the extrapolation to the last anchor. "all" considers all experiments, 10% fits learning curves up to 10% of the total dataset, etc. If two are connected by a red line, the pairwise test did not find significant differences between their performance. If a model doesn't have a line to the axis it's significantly different from all others. Numbers indicate rank (lower is better). (Color figure online)

`lin2`, `exp2` are definitely not suitable for extrapolation, and to a lesser extend we can also already rule out `pow2`.

Since these plots are very aggregated and do not say anything about significance, we visualize ranks in Critical Diagrams (CD) [7] following the approach of [16] in Fig. 5. These figures show the ranking of the different models (lower values are better) while statistically not significantly different ones are tied with red bars. We use the same statistical tests as [3,16], Friedman's test [11] to judge if there are significant differences between curve models, and we use pairwise Wilcoxon signed-rank tests [36] to compare pair of curve models with Holm's alpha correction [15] following [16] with $\alpha = 0.05$.

The obtained ranks from the Friedman tests are given in Table 2. We partition all curve fitting experiments into 6 sets: "all", where all experiments are used, "5%", where all anchors up to exactly 5% of the training set size are used for fitting, "10%", "20%", "40%" and "80%". For all these partitions and performance measures we find significant differences from the Friedman test.

Table 2. Summarized ranks according to the Friedman test for the squared loss to extrapolation to the last anchor. "all" considers all experiments, 10% fits a learning curves up to 10% of the total dataset, etc. Blue/larger numbers means a worse rank, yellow/smaller number indicate a better rank. The last row gives the rank for the MSE on the train anchors over all experiments.

curve	last1	pow2	log2	exp2	lin2	ilog2	pow3	exp3	vap3	expp3	expd3	logp3	pow4	mmf4	wbl4	exp4
all	6.34	10.32	9.58	14.89	14.10	7.95	7.26	8.75	8.66	7.79	8.18	6.86	5.95	6.08	6.58	6.71
5%	7.27	9.28	8.72	14.68	13.93	6.82	7.41	8.23	9.48	7.24	7.88	7.25	6.48	6.91	7.28	7.14
10%	7.14	9.95	9.10	15.02	14.17	7.16	7.09	8.57	9.03	7.33	8.13	6.72	6.19	6.55	6.96	6.87
20%	6.77	10.46	9.48	15.05	14.15	7.73	7.00	8.98	8.52	7.52	8.47	6.50	5.98	6.10	6.58	6.70
40%	6.12	10.74	9.79	14.84	13.92	8.45	7.19	9.30	8.41	7.88	8.54	6.59	5.69	5.77	6.33	6.44
80%	5.16	11.02	10.24	14.40	13.62	9.29	7.34	9.48	8.20	8.27	8.53	6.84	5.65	5.59	6.12	6.25
trn	15.58	11.89	11.51	13.65	12.85	11.25	7.67	8.60	7.04	7.59	7.02	7.60	3.28	2.64	3.76	4.07

In Fig. 5 the critical diagrams are shown (the ranks in this Figure correspond with those in Table 2). However, due to space limitations, the figure only shows the ranks for extrapolations to the *last* anchor; the supplement also contains CD plots for curve fitting and extrapolation performance across the whole upcoming curve (all test anchors) and interpolation on the train anchors. The results for extrapolating to all test anchors is quite similar to extrapolation to the last anchor, but differences between models are larger for the latter.

In accordance with the previous observations, we again see that `exp2` and `lin2` indeed do not perform well, and the performances `log2` and `pow2` are only slightly better. That is unsurprising, since `pow2` and `exp2` cannot converge to a non-zero error rate, and `log2` diverges in the limit. Surprisingly, `exp3`, which can model non-zero error in the limit, obtains a similar ranking and gets progressively worse with more training anchors, indicating it is indeed not suitable. These models are followed by `vap3`, `expp3`, `expd3` that obtain similar ranks.

There is also a group that often tie and attain, generally, much better ranks. `pow3`, `ilog2`, `logpower3` especially works well for small sample sizes and tie often. Nevertheless, for larger sample sizes especially `ilog2` and `logpower3` deteriorate a lot. If little anchors are used it is hard to distinguish performances. However, if more than 20% of the data is used, a new group of overall best model appears, which are `mmf4`, `wbl4`, `pow4`, `exp4`. These models tie often according to the pairwise tests, but `mmf4` and `wbl4` together significantly outperforms the others, especially for 20%-80%. Finally, the baseline `last1` doesn't perform well for less than 20% data, but improves its rank the larger the percentage, and even wins significantly from all others at 80%. This is expected since the curve often plateaus for large sample sizes and no fitting is done. For 'all' `mmf4` performs significantly better than all others with `wbl4` second and `last1` third.

4 Discussion and Conclusion

The learning curve database (LCDB) provides the first extensive collection of learning curves easily accessible and readily deployable for the in-depth study

of learning curves behavior. Importantly, the database provides probabilistic predictions that allows the study a wide range of standard and specialized performance metrics. The most important aspects of these learning curves have been utilized in a PyPI package, for easy and fast usage.

Our preliminary study of LCDB already provides some insights: for error-rate learning curves, we found that the large majority of learning curves is, largely, well-behaved, in that they are monotone, convex, and do not show peaking. We also established empirical estimates of the probability that learning curves cross. Furthermore, our curve fitting experiments emphasize the need for more robust fitting. While we have taken some steps to rule out biases from curve fitting issues, our curve fitting results warrant further analysis to rule this out completely. In contrast to other benchmarks that generally extrapolate curves with 2 or 3 parameter power laws, exponentials or logarithmic models, we find that 4 parameter models are quite competitive, with `mmf4` obtaining the best overall results and `wbl4` a close second. Not surprisingly, the amount of anchors used for training seems to influence which curve model performs best. Of course, many research questions remain, which we hope can be successfully addressed with LCDB. For instance, does it make sense to smooth learning curves for model selection or curve extrapolation? Can meta-features [2], like the number of instances, be used to reliably predict (i) whether curves intersect, (ii) if they are monotone or convex, (iii) or which curve model will be accurate? Which non-parametric extrapolation techniques work best and in what case? In how far can training curves support the extrapolation task? Finally, our paper did not touch upon the aspect of training runtimes, which are however also part of LCDB and can be used to develop sophisticated runtime models. These and many additional interesting learning curve problems can now be efficiently investigated.

LCDB is designed to be a continual database that can and will be extended over time and also integrated with other services. Needless to say that we will seek to add new learning curves to LCDB for new datasets. The main challenge here is to assure that the training runtimes will be comparable: while all the data in the initial database has been generated with the same hardware, this is not necessarily the case for upcoming datasets. Addressing this issue, e.g., by normalizing the runtimes based on calibration techniques is an interesting research question in itself. To further improve the exploitation of LCDB, parts of it will be integrated with OpenML.org in the immediate future.

It should be clear that there are also various technical extensions that could be desirable. First and most natural, an extension towards other learning problems like regression would be useful. Next, LCDB should be extended to cover machine learning *pipelines* instead of learners only. This is however a tough undertaking since the space of pipelines is large, and it is not clear how to select a reasonable subset of those; also storing the prediction vectors for an increased number of learners is a logistic issue. Third, an extension of LCDB for monotonically increasing training sets would be great in order to allow analyses for

data acquisition situations instead of only model selection situations as covered currently. In all, we expect that LCDB will be of great use to address many further interesting and valuable research questions concerning learning curves.

References

1. Benavoli, A., Corani, G., Mangili, F.: Should we really use post-hoc tests based on mean-ranks? J. Mach. Learn. Res. **17**(1), 152–161 (2016)
2. Brazdil, P., van Rijn, J.N., Soares, C., Vanschoren, J.: Metalearning: Applications to Automated Machine Learning and Data Mining, 2nd edn. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-67024-5
3. Brumen, B., Cernezel, A., Bosnjak, L.: Overview of machine learning process modelling. Entropy **23**(9), 1123 (2021)
4. Brumen, B., Rozman, I., Heričko, M., Černezel, A., Hölbl, M.: Best-fit learning curve model for the C4.5 algorithm. Informatica **25**(3), 385–399 (2014)
5. Cohn, D., Tesauro, G.: Can neural networks do better than the Vapnik-Chervonenkis bounds? In: Advances in Neural Information Processing Systems, vol. 3, pp. 911–917 (1991)
6. Cortes, C., Jackel, L.D., Solla, S.A., Vapnik, V., Denker, J.S.: Learning curves: asymptotic values and rate of convergence. In: Advances in Neural Information Processing Systems, vol. 6, pp. 327–334. Morgan Kaufmann (1993)
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
8. Domhan, T., Springenberg, J.T., Hutter, F.: Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, pp. 3460–3468. AAAI Press (2015)
9. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Advances in Neural Information Processing Systems, vol. 28. pp. 2962–2970 (2015)
10. Frey, L.J., Fisher, D.H.: Modeling decision tree performance with the power law. In: Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics. Society for Artificial Intelligence and Statistics (1999)
11. Friedman, M.: A comparison of alternative tests of significance for the problem of m rankings. Ann. Math. Stat. **11**(1), 86–92 (1940)
12. Garcia, S., Herrera, F.: An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J. Mach. Learn. Res. **9**(89), 2677–2694 (2008)
13. Gijsbers, P., LeDell, E., Thomas, J., Poirier, S., Bischl, B., Vanschoren, J.: An open source AutoML benchmark. arXiv preprint arXiv:1907.00909 (2019)
14. Gu, B., Hu, F., Liu, H.: Modelling classification performance for large data sets. In: Wang, X.S., Yu, G., Lu, H. (eds.) WAIM 2001. LNCS, vol. 2118, pp. 317–328. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-47714-4_29
15. Holm, S.: A simple sequentially rejective multiple test procedure. Scand. J. Stat. **6**, 65–70 (1979)
16. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.-A.: Deep learning for time series classification: a review. Data Min. Knowl. Disc. **33**(4), 917–963 (2019). https://doi.org/10.1007/s10618-019-00619-1

17. Klein, A., Falkner, S., Springenberg, J.T., Hutter, F.: Learning curve prediction with Bayesian neural networks. In: 5th International Conference on Learning Representations. OpenReview.net (2017)
18. Kolachina, P., Cancedda, N., Dymetman, M., Venkatapathy, S.: Prediction of learning curves in machine translation. In: The 50th Annual Meeting of the Association for Computational Linguistics, pp. 22–30. The Association for Computer Linguistics (2012)
19. Last, M.: Predicting and optimizing classifier utility with the power law. In: Workshops Proceedings of the 7th IEEE International Conference on Data Mining, pp. 219–224. IEEE Computer Society (2007)
20. Leite, R., Brazdil, P.: Improving progressive sampling via meta-learning on learning curves. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 250–261. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30115-8_25
21. Leite, R., Brazdil, P.: Selecting classifiers using metalearning with sampling landmarks and data characterization. In: Proceedings of the 2nd Planning to Learn Workshop (PlanLearn) at ICML/COLT/UAI, pp. 35–41 (2008)
22. Li, L., Jamieson, K.G., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: bandit-based configuration evaluation for hyperparameter optimization. In: 5th International Conference on Learning Representations. OpenReview.net (2017)
23. Li, Y.: An investigation of statistical learning curves: do we always need big data? Master's thesis, University of Canterbury (2017)
24. Loog, M., Viering, T., Mey, A.: Minimizers of the empirical risk and risk monotonicity. In: Advances in Neural Information Processing Systems, vol. 32, pp. 7478–7487 (2019)
25. Loog, M., Viering, T.J., Mey, A., Krijthe, J.H., Tax, D.M.J.: A brief prehistory of double descent. Proc. Natl. Acad. Sci. **117**(20), 10625–10626 (2020)
26. Mohr, F., van Rijn, J.N.: Towards model selection using learning curve cross-validation. In: 8th ICML Workshop on Automated Machine Learning (2021)
27. Mohr, F., van Rijn, J.N.: Learning curves for decision making in supervised machine learning - a survey. CoRR abs/2201.12150 (2022)
28. Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., Sutskever, I.: Deep double descent: where bigger models and more data hurt. In: 8th International Conference on Learning Representations. OpenReview.net (2020)
29. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
30. Perlich, C., Provost, F.J., Simonoff, J.S.: Tree induction vs. logistic regression: a learning-curve analysis. J. Mach. Learn. Res. **4**, 211–255 (2003)
31. Provost, F.J., Jensen, D.D., Oates, T.: Efficient progressive sampling. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 23–32. ACM (1999)
32. van Rijn, J.N., Abdulrahman, S.M., Brazdil, P., Vanschoren, J.: Fast algorithm selection using learning curves. In: Fromont, E., De Bie, T., van Leeuwen, M. (eds.) IDA 2015. LNCS, vol. 9385, pp. 298–309. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24465-5_26
33. Singh, S.: Modeling performance of different classification methods: deviation from the power law. Project Report, Department of Computer Science, Vanderbilt University, USA (2005)
34. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. SIGKDD Explor. **15**(2), 49–60 (2014)

35. Viering, T.J., Loog, M.: The shape of learning curves: a review. CoRR abs/2103.10948 (2021)
36. Wilcoxon, F.: Individual comparisons by ranking methods. In: Kotz, S., Johnson, N.L. (eds.) Breakthroughs in Statistics, pp. 196–202. Springer, Cham (1992). https://doi.org/10.1007/978-1-4612-4380-9_16
37. Zimmer, L., Lindauer, M., Hutter, F.: Auto-PyTorch tabular: multi-fidelity MetaLearning for efficient and robust AutoDL. IEEE Trans. Pattern Anal. Mach. Intell. **43**(9), 3079–3090 (2021)