

Learning Curves for Decision Making in Supervised Machine Learning: A Survey

Felix Mohr^{1*} and Jan N. van Rijn²

¹Universidad de La Sabana, Chía, Cundinamarca, Colombia.

²Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands.

*Corresponding author(s). E-mail(s):
felix.mohr@unisabana.edu.co;

Contributing authors: j.n.van.rijn@liacs.leidenuniv.nl;

Abstract

Learning curves are a concept from social sciences that has been adopted in the context of machine learning to assess the performance of a learning algorithm with respect to a certain resource, e.g., the number of training examples or the number of training iterations. Learning curves have important applications in several machine learning contexts, most notably in data acquisition, early stopping of model training, and model selection. For instance, learning curves can be used to model the performance of the combination of an algorithm and its hyperparameter configuration, providing insights into their potential suitability at an early stage and often expediting the algorithm selection process. Various learning curve models have been proposed to use learning curves for decision making. Some of these models answer the binary decision question of whether a given algorithm at a certain budget will outperform a certain reference performance, whereas more complex models predict the entire learning curve of an algorithm. We contribute a framework that categorises learning curve approaches using three criteria: the decision-making situation they address, the intrinsic learning curve question they answer and the type of resources they use. We survey papers from the literature and classify them into this framework.

Keywords: learning curves, supervised machine learning

2 Learning Curves for Decision Making

1 Introduction

Learning curves describe a system's performance on a task as a function of some resource to solve that task. There can be a pre-defined budget of that resource, limiting the amount of resources that can be spent. In other cases, the goal can be to obtain reasonable results while minimising the spent budget of that resource. Typical types of budgets are the number of *examples* the learner has observed before performing the task or the number of *iterations* or *time* the learner spends in an environment. The performance measure expresses the quality of the obtained model, e.g., error rate or F1 measure. Learning curves are an important source of information for making decisions on the following matters in machine learning:

- *Data Acquisition* determines how many data points should reasonably be acquired to obtain a desired performance. The top right plot in Fig. 1 visualises a scenario where we have already observed performance up to a certain amount of data (the blue learning curve). We can extrapolate this and make a prediction of what the performance would be if more data was available, i.e., the value of the orange extrapolation at different vertical pink lines in the figure (see, e.g., Last, 2009; Weiss and Tian, 2008).
- *Early Stopping* of training a model. If we are committed to some specific learner (a learning algorithm *and* its hyperparameters), we might want to minimise the training time (John and Langley, 1996; Provost et al, 1999) or avoid over-fitting (Bishop, 1995; Goodfellow et al, 2016). The middle right plot in Fig. 1 visualises a scenario where we have already observed performance up to a certain amount of budget, and based on the progression of the learning curves on recent iterations, a decision can be made whether to continue the learning or terminate it.
- *Early Discarding* in model selection. If we want to *select* from various models, we want to stop the evaluation of a candidate when we are reasonably certain that it is not *competitive* to the best-known solution (Domhan et al, 2015; Mohr and van Rijn, 2023; Swersky et al, 2014). The bottom right plot of Fig. 1 visualises a scenario where we have already observed performance up to a certain amount of budget (the blue learning curve) and already an incumbent performance obtained by an earlier configuration (horizontal dashed pink line). By using learning curve extrapolation techniques, we can determine whether the current configuration can surpass the incumbent configuration; if not, discarding the current training process early (as in the case shown) would be justified.

Many techniques with varying complexity and required resources have been proposed to address either of these problems. The complexity ranges from approaches that simply recognise whether an already observed part of a learning curve has converged (Bishop, 1995; Provost et al, 1999) to the creation of parametric learning curve models, which capture a belief model for the behaviour of various learners at any possible budget (Klein et al, 2017b). While simple approaches may only rely on the observations made so far for a learner

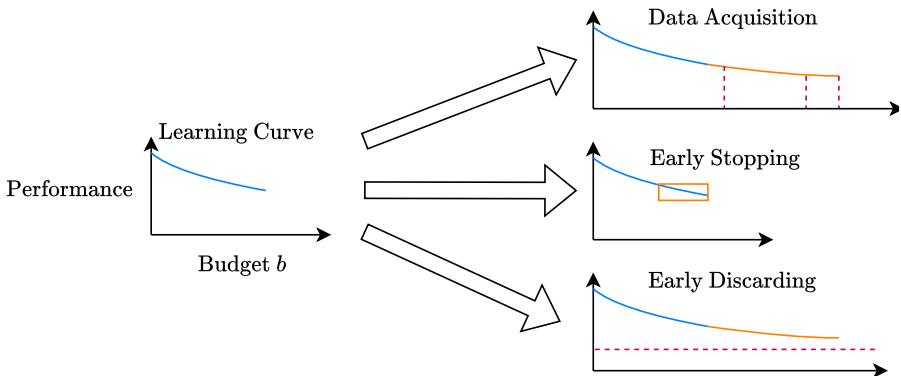


Fig. 1: The three types of decision-making situations in which learning curves are typically used. The x -axis of each figure represents the budget in the applicable unit, and the y -axis represents the performance.

on the dataset of interest, more complex approaches may rely on additional resources such as learning curves or features of other datasets (see, e.g., [Leite and Brazdil, 2010](#)) or learners (see, e.g., [Chandrashekaran and Lane, 2017](#)) or both (see, e.g., [Ruhkopf et al, 2023](#)).

Our contribution is a unified framework of the usage of learning curves for decision making in machine learning and an extensive review of the literature of approaches that fall within this framework. This framework categorises the existing literature along the following three axes:

1. The type of decision-making situation, i.e., whether it is used to make decisions about data acquisition, early stopping, or early discarding. See Sec. 4.1 for more details.
2. The type of technical question that can be answered with an approach, e.g., some approaches can only answer the binary question whether a model has converged, whereas other approaches are able to answer questions about the behaviour of any part of the learning curve. See Sec. 4.2 for more details.
3. The data resources that are used to model the learning curve. For example, in some cases, data from different algorithms on the same dataset is being used, whereas in other cases, data from the same algorithm on other datasets. See Sec. 4.3 for more details.

We perform an extensive literature survey in which we categorise published learning curve extrapolation models along the various axes of this framework. This literature survey is subject of Sec. 5, which is then summarised in Table 1. We focus specifically on supervised machine learning, in which learning curves describe the predictive performance of a model produced by a learning algorithm either as a function of the number of training instances or of the time or iterations spent for learning on a given dataset. We explicitly exclude learning

4 Learning Curves for Decision Making

curves that describe the performance of a learning agent in an environment over time, i.e., the learning curve of an agent in a reinforcement learning setup (Waltz and Fu, 1965). Similarly, we briefly contrast learning curves to other performance curves, such as active learning curves, feature curves, and capacity curves, and explain why we consider these out of scope for the literature review. Still, we aim to survey exhaustively the literature that introduces approaches that use learning curves in supervised learning.

Contributions:

Our contributions are the following.

- We present a unified framework of the usage of learning curves for decision making in machine learning and an extensive review of the literature on approaches that fall within this framework. This framework contains three axes, i.e., (i) the type of decision-making situation, see also Fig. 1 (ii) the type of question that can be answered, see also Fig. 10 and (iii) the data resources that are used to model the learning curve, see also Fig. 11. While the first axis of this framework is also used in other literature (see, e.g., Viering and Loog, 2023), to the best of our knowledge, the other two axes have not yet been explicitly identified.
- We conducted a literature survey in which we categorise methods presented in the literature along the various axes of this framework. Sec. 5 lists all these methods (where each subsection represents a type of question being answered), and Table 1 overviews all methods along the three axes of our framework.
- Based on the framework, we identify unexplored routes for further research. Most notably, we note that there is a mismatch between the research questions being answered and the learning curve modelling method being used; in many cases, a high-level modelling technique is used to answer a low-level question. We speculate that matching the level of the question being answered with the appropriate level of the modelling technique can further improve the obtained results.

Relation to other literature reviews on learning curves:

Another prominent literature review that centres around learning curves is the highly complementary work by Viering and Loog (2023), which has been developed in parallel. While both works have identified the three types of decision-making situations that are referred to in the literature, Viering and Loog (2023) survey more theoretical work that analyses the shape of learning curves, whereas this work surveys work that is more oriented towards methods that extrapolate learning curves, thereby supporting the data scientists in various decision-making situations.

Structure:

This paper is structured into three main parts. Sec. 2 presents relevant background knowledge on learning curves, including formal definitions and the

terminology relevant for the remainder. Sec. 3 presents relevant important concepts that relate to how learning curves are generally modelled. Sec. 4 contains our main contribution by introducing our framework for categorising methods that utilise learning curves for decision making in supervised learning. Following this framework, Sec. 5 exhaustively reviews approaches that explicitly or implicitly answer questions related to learning curves to make or recommend decisions in the context of supervised machine learning. Sec. 6 concludes our findings. Finally, Appendix A presents a table that overviews the most critical notation used throughout this paper.

2 Background on Learning Curves

This section gives a conceptual background on learning curves. It first provides an idealised formal definition in Sec. 2.1 followed by a definition of *empirical* learning curves in Sec. 2.2 that can be computed in practice. The concept of *utility* curves is introduced in Sec. 2.3. Sec. 2.4 introduces important terminology such as *anchor points*, *limit performance*, and the *saturation point*. Finally, Sec. 2.5 contrasts the learning curves covered in this survey with other types of performance curves used in machine learning.

2.1 Sample-Wise and Iteration-Wise Learning Curves

We consider learning curves in the context of supervised machine learning. Formally, in the supervised learning context, we assume 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 as \mathcal{D} the set of all possible datasets. A *learning algorithm* is a function $a : \mathcal{D} \times \Omega \rightarrow H$, where $H = \{h \mid h : \mathcal{X} \rightarrow \mathcal{Y}\}$ is the space of hypotheses and Ω is a source of randomness.

Note that learning curves can also be considered in other machine learning setups. In fact, learning curves appeared first in reinforcement learning (Waltz and Fu, 1965) and have also been used for unsupervised learning (Meek et al, 2002). However, to give this survey focus, we consider learning curves for supervised learning.

The performance of a hypothesis is typically expressed as *risk*, which is also often called *out-of-sample error*:

$$\mathcal{R}_{\text{out}}(h) = \int_{\mathcal{X}, \mathcal{Y}} \text{loss}(y, h(x)) d\mathbb{P}_{\mathcal{X} \times \mathcal{Y}}. \quad (1)$$

Here, $\text{loss}(y, h(x)) \in \mathbb{R}$ 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 a joint probability measure on $\mathcal{X} \times \mathcal{Y}$ from which the available dataset d has been generated. As such, the out-of-sample error represents the weighted summed error that hypothesis h makes on all possible instance-label pairs, weighted by their probabilities.

6 Learning Curves for Decision Making

The performance of a *learning algorithm* is simply the performance of the hypothesis it produces. In contrast to the performance of a hypothesis, the performance of a learner depends on its input, i.e., on the data provided for learning. The average performance of learner a for a number n of training examples can then be expressed as

$$\mathcal{C}(a, n) = \int_{\omega \in \Omega, d_{tr} \in \mathcal{D}, |d_{tr}|=n} \mathcal{R}_{out}(a(d_{tr}, \omega)) d\mathbb{P}_{\mathcal{X} \times \mathcal{Y}} d\mathbb{P}_\Omega, \quad (2)$$

where $d_{tr} \in \mathcal{D}$ is the dataset of size n used to induce a model using learner a . It is generally assumed that d is a collection of i.i.d. samples from $\mathbb{P}_{\mathcal{X} \times \mathcal{Y}}$.

When we consider Eq. (2) as a function of the number of training samples for a fixed learner a , we obtain the *sample-wise curve* of learner a . That is, the sample-wise curve is the function $\mathcal{C}(a, \cdot) : \mathbb{N} \rightarrow \mathbb{R}$; so it is a *sequence* of performances, one for each training size. Fig. 2 (left) visualises this by means of the green line and compares this to two other types of learning curves with the error rate as the loss (see Sec. 2.5).

Alternatively, many learning algorithms implement an iterative internal optimisation process, which allows describing the learning progress over time or a number of iterations. For example, neural network training produces a new hypothesis after every batch or epoch, ensemble learners like bagging or boosting produce a new hypothesis after every added ensemble member, and support vector machine optimizers yield updated attribute or instance coefficients in iteration. In the formal framework, a learner can be seen more generally as a function $a : \mathcal{D} \times \Omega \rightarrow H^+$ that maps a dataset to a *sequence* of hypotheses, one for each of its iterations. The above error function for learners can then be written as

$$\mathcal{C}(a, n, t) = \int_{\omega \in \Omega, d_{tr} \in \mathcal{D}, |d_{tr}|=n} \mathcal{R}_{out}(a(d_{tr}, \omega)_t) d\mathbb{P}_{\mathcal{X} \times \mathcal{Y}} d\mathbb{P}_\Omega, \quad (3)$$

Here, t expresses some budget, for example, time or a number of iterations over the dataset, often expressed in epochs.

Based on this notion, the *iteration-wise curve* of a learner a is defined for a *fixed* dataset size n (often between 70% and 90% of the available data) and is then the function $\mathcal{C}(a, n, \cdot) : \mathbb{N} \rightarrow \mathbb{R}$. Fig. 2 (right) visualises an example of two iteration-wise curves. It can be seen that these iteration-wise curves are also influenced by the number of samples. The sample-wise curve in this example (visualised by the dashed line) is assumed to utilise the maximal number of iterations. Examples of such learning curves occur above all in the analysis of deep learning models (Domhan et al., 2015; Goodfellow et al., 2016).

The two types of learning curves seem to be related and indeed look similar when visualised, but they have different semantics. The crucial difference is that iteration-wise curves are usually visualized for a fixed and finite number of training samples (expressed by n), no matter how large t becomes. In fact,

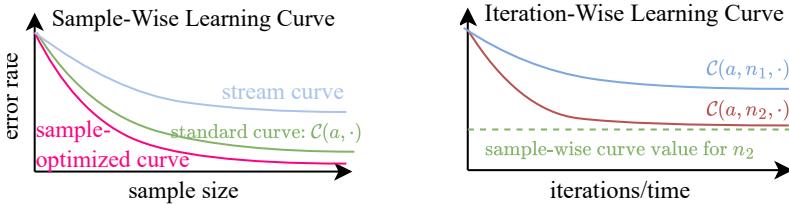


Fig. 2: Left: (Standard) sample-wise curve (green) for a single learner on a particular data source together with learning curves under sample optimisation (pink) and learning curves on streams (blue). Right: iteration-wise curves of a single learner on a particular data source for two different dataset sizes $n_1 < n_2$.

since iterative learners typically automatically stop the learning process as soon as no progress is observed, it often holds that $\mathcal{C}(a, n) = \lim_{t \rightarrow \infty} \mathcal{C}(a, n, t)$. But this is not necessarily the case, specifically if an algorithm stops early (in the iteration-wise curve), e.g., to avoid over-fitting (as the neural network in Fig. 3). Note that as t grows, each training instance is considered an unlimited number of times; hence, iteration-wise curves show how much the learner can make out of a constant number of training instances. Instead, sample-wise curves show the performance of the learner as the number of *examples* grows. The latter typically means, for an infinite input space, that the *information basis* available to the learner is growing strictly bigger, while the information is constant in the case of iteration-wise curves.

Note that while the learning success can be expressed in a metric that ought to be maximised or minimised, in this paper we assume for simplicity that they are to be minimised. This is why the performance is expressed through a *loss* such as the error rate. In this case, learning curves are (usually) *decreasing*. However, more generally, one can also be interested in *increasing* learning curves, e.g., when considering accuracy or the F1 measure. Since learning curves can be simply mirrored at the x-axis, every approach discussed in this paper is applicable to both increasing or decreasing learning curves. For simplicity, this survey assumes that lower performance values are better (error rate, log-loss/cross-entropy, Brier score, mean-square-error, etc.).

2.2 Empirical Learning Curves

The above definitions of learning curves are purely theoretical. This is because we cannot evaluate equations (1–3) in practice. First, the out-of-sample error \mathcal{R}_{out} , i.e., Eq. (1) cannot be computed in practice since the measure $\mathbb{P}_{\mathcal{X} \times \mathcal{Y}}$ is unknown. Relying on this error, the learning curve values cannot be computed either. The necessity to average over the oftentimes uncountable set of all possible train sets can add additional problems.

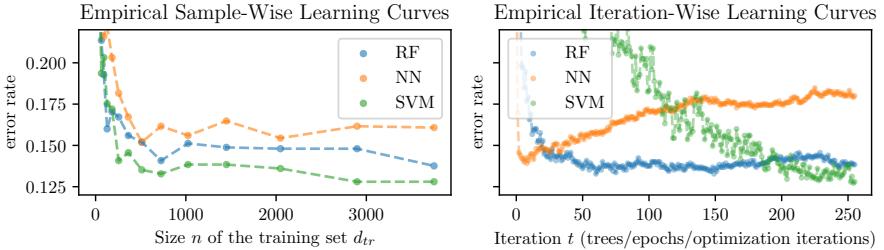
8 *Learning Curves for Decision Making*

Fig. 3: Empirical learning curves for the waveform dataset. Left: sample-wise curves at different training set sizes up to 80% of the data. The remaining 20% are used to compute the error. Right: iteration-wise curves with one entry for each forest size (RF), epoch (NN), or optimization iteration (SVM) when a fixed set of 80% of the available data is used in each iteration for training and the rest to compute the error.

To compute learning curves in practice, we rely on empirical *estimates* of the above quantities. We estimate the out-of-sample error by the *internal error*:

$$\mathcal{R}_{in}(h) = \frac{1}{|d|} \sum_{(x,y) \in d} loss(y, h(x)), \quad (4)$$

where d is the dataset used for assessing the performance of hypothesis h . Note that the dataset d may or may not contain instances used to create the hypothesis h . In most practical applications, dataset d consists of instances that have not been seen during the creation of hypothesis h (i.e., the test set). However, in theory, the internal error can also be estimated based on the train set d_{tr} or a combination of instances from the train set and test set.

We consider an *empirical learning curve* any set of estimates of a true learning curve for different sizes or iterations. We can use various estimation procedures to estimate the performance using a given training size, such as using a regular holdout set or cross-validation. The latter leads to various estimates, and averaging over these estimates yields an estimate of the sample-wise curve in Eq. (2) at size n . To obtain an empirical estimate of the iterative learning curve in Eq. (3), we do the same except that we stop the learning algorithm after t iterations.

Fig. 3 shows empirical learning curves for a Random Forest (RF), a Neural Network (NN) with 100 neurons in a hidden layer, and a support vector machine (SVM) with RBF kernel on a concrete and widely used benchmark dataset (the waveform dataset). The empirical curves are the scatter points; the lines here are only a visual aid. The error rate is here obtained from a single validation fold, i.e., without averaging, which explains the rather unsMOOTH behavior. The iteration-wise curves were created using 80% of the data for training.

Since empirical learning curves are the only way to gain insights about true learning curves, quite some studies have been published with the sole goal of

sharing empirical learning curves with the community and thereby improving the understanding of how they behave. Perlich et al (2003) contrast the learning curves on decision trees and logistic regression on different datasets. Notably, the authors also *compare* learning curves, e.g., they report whether one curve is below the other (dominates it) on all considered training set sizes or whether the two learning curves cross. Several other studies report learning curves for specific learners. Ng and Jordan (2001) compare logistic regression and naive Bayes. Mørch et al (1997) conduct a study similar to the one by Perlich et al (2003) to compare linear vs. non-linear classifiers on a smaller scale. Recently, a number of learning curve databases have been published, i.e., learning curves of different network architectures on typical image classification datasets (Bornstein et al, 2020; Dong and Yang, 2020; Siems et al, 2020), learning curves of different machine learning algorithms on tabular data (Mohr et al, 2022), and for mixtures of these tasks (Eggensperger et al, 2021; Pfisterer et al, 2022).

Empirical studies of this type do not answer generalising questions about learning curves but rather report experimental results. This is different from contributions in which certain model assumptions are made and data is compared to those models, e.g., with the aim to compute the goodness of fit of that model. In Sec. 3, we briefly discuss some of such models.

2.3 Utility Curves

The concept of learning curves can be further generalised to a *utility curve* (Last, 2007, 2009; Weiss and Tian, 2008). The utility usually involves a trade-off between the performance and the computational cost of training a model. The specific details can be different per task. The utility is connected to the learning curve in so far as the utility is also a function of the budget and is directly influenced by the predictive performance. Therefore, one could argue that the utility curve \mathcal{U} is obtained by passing the learning curve to the utility function alongside other parameters that influence the utility, most notably the cost of acquiring new instances and the cost to train a model on the respective dataset size. The learning curves associated with utility costs are visualised in Fig. 4 (orange) and compared to a normal learning curve. Assuming there is a linear cost associated with further training the classifier, we can see that the utility curve (which makes a trade-off between performance and cost) peaks at a given point, and deteriorates afterwards.

2.4 Terminology of Learning Curves

While this survey is not primarily about the shapes of learning curves, the shapes of learning curves play an important role when using them to make decisions. Hence, we consider it necessary to convey some of the most important insights about the basics of the shapes of learning curves. However, we refer to a recent exhaustive survey on the shapes of learning curves (Viering and Loog, 2023) for details on this topic. Fig. 5 visualises some important concepts.

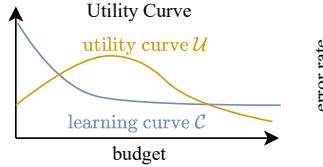


Fig. 4: A utility curve with the corresponding learning curve.

Anchor Points

In this survey, we adopt the term *anchor* to refer to a point for which the empirical learning curve carries a performance estimate. There is no established name for such points in literature. They are called *sample sizes* in (John and Langley, 1996; Leite and Brazdil, 2007; Provost et al, 1999), and those authors refer to a *collection* of such samples sizes as a *schedule* (Figueroa et al, 2012; John and Langley, 1996; Meek et al, 2002; Provost et al, 1999). However, for iteration-wise curves, the term ‘sample’ is misleading because the curve plots performance against the number of times all instances (out of a fixed set) are presented to the learner, which is not the same as sample size. Besides, the term sample size is quite overloaded in the context of machine learning, because this field deals with various types of samples in different contexts, e.g., train and validation samples, etc. Another terminology observed sometimes is the one of *sample landmarks* (Fürnkranz and Petrak, 2001; Leite and Brazdil, 2005). However, this term is also slightly confusing, since landmarks are generally understood as the performances of cheap-to-evaluate learners, which was also the motivation for this terminology by Fürnkranz and Petrak (2001). A less used terminology is the term *anchor* (Kielhöfer et al, 2024; Kolachina et al, 2012; Mohr and van Rijn, 2021, 2023), which is not ambiguous in the machine learning context and captures the idea that analysis is based on some selected points. It serves well to immediately create an association with a particular size of a sampled training data set or a number of visited instances that is used in the context of building an empirical learning curve.

Throughout this paper, we formally use the symbol b to refer to an anchor. Thereby, we abstract away from sample sizes n or iteration t . In other words, the symbol b is used to indicate points on both sample-wise curves or iteration-wise curves, and it should be clear from the context which one is meant (if the difference is relevant).

For example, in Fig. 3 above, we have the following anchors. For the sample-wise curve, a geometric schedule with $n \in \{\lceil 2^{i/2} \rceil \mid i \in \mathbb{N}\}$ is chosen. In this concrete case, the anchors are $\{1, 2, 3, 4, 6, 8, 12, 16, 23, 32, 46, 64, 91, 128, 182, 256, 363, 512, 725, 1024, 1449, 2048, 2897, 3750\}$. In the iteration-wise curve, every iteration is used as an anchor, so $t \in \{1, \dots, 200\}$.

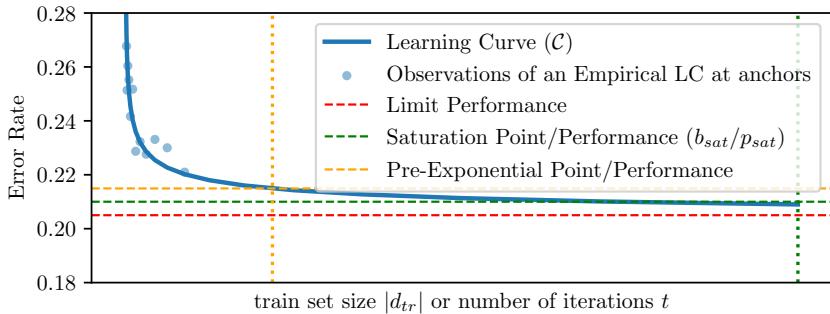


Fig. 5: Concepts related to a learning curve. Red: Limit performance. Green: Saturation point b_{sat} (vertical line) and saturation performance p_{sat} (horizontal line). Orange: Pre-exponential point (vertical line) and pre-exponential performance (horizontal line). The curve plateaus at a performance of 0.2; the plateau is not visualized here to emphasise the difference between the pre-exponential point and saturation point.

Limit Performance

It is generally assumed that learning curves *converge* to some value. In the case of iterative learning curves, there are sometimes oscillations in the curve, but even in such cases, the curve usually converges to some (in those cases, typically a bad) value eventually. We are not aware of a particular term that is used to describe the score to which a learner converges. Cortes et al (1994) describes the *limiting performance* or the *asymptotic performance* of the *data*; i.e., it is not the property of a particular learner but the best achievable performance among all learners under consideration (even though only tested with two model types in the paper). In this paper, we adopt the term *limit performance* of the learner (on a fixed number of training instances in the case of iteration-wise curves), and we denote this performance as p_{lim} .

Saturation Point

Intuitively, the *saturation point* is the anchor after which the performance converges. That is, the anchor after which all values are in a distance of less than some pre-defined and typically very small ε . In early works, this point has been called *stopping point* (Figueroa et al, 2012; Leite and Brazdil, 2003, 2004; Meek et al, 2002; Provost et al, 1999). Provost et al (1999) characterise this point as follow: “Models built with smaller training sets [than s_{min}] have lower accuracy than models built with from training sets of size s_{min} , and models built with larger training sets have no higher accuracy.” In the context of those works, namely progressive sampling, the term stopping point makes sense because they progressively sample until they reach the convergence region, and then stop the sampling procedure. In the absence of such a mechanism, the term appears a bit odd. The notion of saturation in the context of learning

curves was proposed by [Tomanek \(2010\)](#) and seems appropriate since the limit performance has not been reached, but it has *almost* been reached. The curve is saturated up to a mistake of ε .

We will denote the saturation point itself as b_{sat} , emphasising that the saturation point is an anchor. The *saturation performance* is simply the value of the learning curve at that point. Both are defined in the context of a fixed learner a . Hence, we can write the saturation performance as $p_{sat} := \mathcal{C}(a, b_{sat})$ in the context of a sample-wise curve or $p_{sat} := \mathcal{C}(a, n, b_{sat})$ in the context of an iteration-wise curve.

Pre-Exponential Point

A related concept is the pre-exponential point and, correspondingly, the pre-exponential performance. The saturation point may be expensive to reach in the sense that a lot of training data is necessary to obtain the saturation performance. We call the smallest anchor point for which an increase by a factor of q leads to a performance improvement of less than some δ , i.e., $\mathcal{C}(a, n) - \mathcal{C}(a, q \cdot n) < \delta$. Reasonable candidates for q can be 2 or 10, while candidates for δ can be 0.01 or 0.001 if the metric is the error rate. Its semantic is that from the pre-exponential point on, one needs more than q^k (i.e., an exponentially increasing number of) training samples or iterations to improve by a low margin of $k\delta$. Correspondingly, the pre-exponential performance is the performance that can be obtained by a comparably small anchor point.

Utility-Based Stopping Point

The *utility-based stopping point* is the point at which the acquisition of further data points has a negative impact on the *utility* of the data analysing entity. Therefore, this concept is associated with utility curves. The utility-based stopping point is not related to the saturation point but, if at all, rather to the pre-exponential point; we denote it as b_{sat}^u . In Fig. 4, this is the peak of the yellow curve.

Plateau

The right-sided open interval bounded by the saturation point from the left is consistently called the *plateau* of the curve. However, a curve can also have *intermediate* plateaus, i.e., intervals of (almost) constant performance without being the final plateau.

Well-behaved Learning Curves

To our knowledge, the notion of a *well-behaved* learning curve is first used by [Provost et al \(1999\)](#). A learning curve is said to be well behaved if its slope is monotonically non-decreasing (for error-based learning curves). An even stricter criterion demanding *convexity* of the curve, which implies monotonicity, has been introduced recently by [Mohr and van Rijn \(2021, 2023\)](#). The property of being well behaved is one of the *true* learning curve. The (linear interpolation of an) empirical learning curve can often violate this condition,

specifically when the number of validations conducted at the anchors is small or when the learning curve has reached a plateau.

While it is known that not all learning curves are well behaved (Loog and Duin, 2012; Loog et al, 2019), empirical studies suggest that such curves are rather an exception and that most *sample-wise curves* are well behaved. Learning curves that are not well behaved are found above all in the context of deep learning, where a *double descent* or *peaking* phenomenon can be observed (at times) for both sample-wise curves and iteration-wise curves (Nakkiran et al, 2020); the effect was also observed decades ago for other learners (Vallet et al, 1989). However, extensive empirical studies have shown that most sample-wise curves are even *convex* (Mohr and van Rijn, 2023) and hence well behaved. Some recent works suggest that potential ill behaviour can be mitigated by appropriate configuration or wrapping of learners (Mhammedi and Husain, 2021; Nakkiran et al, 2021; Viering et al, 2020).

2.5 Relation to Other Types of Performance Curves

We briefly discuss the relation of learning curves to other types of curves and problem settings. These curves are fundamentally different from learning curves, and therefore, a more detailed coverage is beyond our scope.

2.5.1 Learning Curves in Active Learning

Active learning is a setting where the data scientist can acquire the label of arbitrary instances (Settles, 2009) and hence can actively increase the training set. On a concrete data source with a concrete initial dataset, a specific active learning strategy creates a *deterministically* extended dataset for any arbitrary anchor. While this allows drawing a curve that plots performance against the number of training instances, this curve is not the one described in Eq. (2), because the datasets are not sampled i.i.d. from $\mathbb{P}_{\mathcal{X} \times \mathcal{Y}}$ but dictated by the active learning strategy; they are *sample-optimised*. Fig. 2 displays how this theoretically relates to the normal sample-wise curve. We do not consider this type of learning curve in this survey.

2.5.2 Learning Curves under Optimal Class Distribution

Similarly, we obtain such a biased learning curve if we do not preserve the *class* distribution. Weiss and Provost (2003) have shown that it can be advantageous to over-sample instances of a minority class if they occur substantially more seldom than instances of a majority class. One can then ask for the best class distribution for a certain anchor. Similar to the active learning case, this creates a new distribution of datasets that does not coincide with $\mathbb{P}_{\mathcal{X} \times \mathcal{Y}}$ anymore. If we optimise over the class distribution at each anchor, we obtain a curve with the same axis labels as a sample-wise curve but a different semantics (and most likely different values). For the case of two classes, this type of learning curve is obtained by taking the budget-wise maximum of a performance *surface* as proposed by Forman and Cohen (2004).

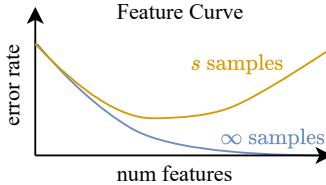


Fig. 6: Feature curve example for a single learner, once for a fixed and finite dataset size and once for an infinite dataset size.

2.5.3 Learning Curves on Data Streams

Another type of learning curve that violates the implicit assumptions made in Eq. (2) is obtained when learning from *data streams*, a scenario in which training instances are coming in sequentially and need to be processed under strict time and memory constraints (Bifet et al, 2018). Incremental learners like Hoeffding trees (Domingos and Hulten, 2000) and models induced by stochastic gradient descent are natural solutions to this problem domain. When applied to a data stream, these algorithms explicitly *forget* an instance once it has been processed. This is not only to free memory but also to address the problem of *concept drift*, i.e., the fact that $\mathbb{P}_{\mathcal{X} \times \mathcal{Y}}$ changes over time. In such a case, we do not have that $\mathcal{C}(a, n) = \lim_{t \rightarrow \infty} \mathcal{C}(a, n, t)$ but rather that $\mathcal{C}(a, n) = \mathcal{C}(a, n, n)$, where the learner is updated at every incoming training instance, and each of the n instances was considered exactly once; for example, similar to training a neural network for one epoch with batch size 1. While this produces a kind of sample-wise curve, the result is clearly different from the learning curve received in the normal batch setting. Fig. 2 displays how these type of curves theoretically relate to the normal sample-wise curves.

Even though syntactically equivalent to a sample-wise curve, data stream learning curves are substantially different and need different treatment. Due to the (potential) concept drift over the time dimension, the i.i.d. assumption is not guaranteed (da Costa et al, 2016). From a theoretical viewpoint, extrapolating the learning curve over time to meaningfully predict future behaviour becomes impossible without the i.i.d. assumption. Due to its special nature, the data stream setting is beyond our scope.

2.5.4 Feature Curves

Learning curves always consider a fixed number of features. Instead, one can fix the number of training instances and consider the performance as a function of the number of features. This yields so-called *feature curves* (Hughes, 1968; Viering and Loog, 2023).

Defining meaningful feature curves is conceptually more difficult than learning curves because of the importance that single features can have. For

simplicity, consider only sample-wise curves for this comparison. In such learning curves, every point n is associated with the expected performance when using n training instances. These n training instances are assumed to be drawn independently and identically distributed from the underlying distribution. According to the aforementioned definitions of a learning curve (per Eq. 2 and Eq. 3), there is no notion of a more informative instance (even though this notion clearly exists in the field of *active learning*). In particular, the *order* in which instances are drawn is irrelevant. However, in the context of features, some features are often more informative than others (and again, other features that have not been measured may be even more informative). Therefore, in order to get an adequate overview, feature curves require some kind of averaging over all possible sets of features of a fixed size that can be formed from a base set of (available) features, as done by Hughes (1968). The fact that some features might be more important than other features makes it hard to model and extrapolate feature curves, as there is no reasonable set of assumptions to build these models on. Fig. 6 displays two theoretical examples of feature curves. When having a finite number of samples, the performance of these curves will, in the limit (when more features are added), deteriorate due to the curse of dimensionality. Of course, when having infinite samples, the performance of such feature curves will go to perfect performance.

Note that feature curves and learning curves can be integrated. For example, Strang et al (2018) look at the combination of the number of instances and the number of features. Since the effect of the number of features and the number of instances on the overall performance is clearly not independent, considering both together is sensible. At the same time, due to the ambiguous semantics of feature curves already discussed above, using such combined curves is not necessarily straightforward for decision making, and we are not aware that the combined curve has been used for decision making so far.

2.5.5 Capacity Curves

Cortes et al (1994) introduce a curve that plots the performance of a configurable learner as a function of the complexity of its instantiation. For example, the learner could be a neural network, and the complexity would then be the number of hidden layers. In doing this, a fixed dataset size is assumed. In that paper, this type of curve has no specific name, but we dub it the *capacity curve* because they plot the performance as a function of capacity. Fig. 7 displays examples of capacity curves.

Capacity curves are interesting from a theoretical viewpoint as they allow us to analyse the intrinsic noise level of the given *data*. More precisely, one can ask for the performance of a learner of some complexity level on, perhaps, an infinite number of data points. If this value can be computed for every complexity level, then we obtain a performance curve over the model complexity. If the number of data points is large enough, this curve can be assumed to be monotonically decreasing. If we have a maximally flexible learner (such as a neural network) that can, in principle, assimilate any function, then the curve

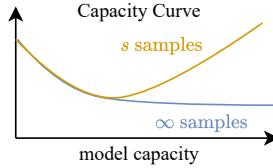


Fig. 7: Capacity curve example for a learner class whose complexity can be increased (e.g., a neural network), once for a fixed and finite dataset size and once for the theoretical case of an infinite dataset size.

will converge towards the *intrinsic noise* of the data. That is, no learner can improve over that performance.

2.5.6 Curriculum Learning

Curriculum learning is a paradigm inspired by human learning strategies, particularly how humans learn complex tasks by gradually increasing the difficulty of the examples they are exposed to (Wang et al, 2022).

In curriculum learning, instead of randomly presenting training instances to the model, instances are presented in a meaningful order, typically from simpler to more complex sets of training instances. This can help the model learn more effectively and converge faster by initially focusing on easier instances that are simpler to learn and gradually introducing more difficult instances.

Learning curves related to curriculum learning come with all sorts of novel challenges, such as interpreting the learning curve in case more complex test instances are provided. Therefore, it is hard to make assumptions about what a well-behaved curriculum learning curve would look like. For this reason, we consider curriculum learning to be out of scope.

3 Modelling a Learning Curve

A learning curve *model* is a characterisation of the *true* learning curve derived from an *empirical* learning curve. The empirical learning curve is the result of sampling from a *stochastic process* that underlies *noise* stemming from randomness in data splits and the learning algorithm itself. It is typically assumed (Domhan et al, 2015; Figueroa et al, 2012; Klein et al, 2017a,b; Mohr and van Rijn, 2023; Swersky et al, 2014) that, for any learner a and any budget $b \in \mathbb{N}$, this stochastic process follows the distribution

$$f(a, b) \sim \mathcal{N}(\mu_{a,b}, \sigma_{a,b}^2) = \mu_{a,b} + \mathcal{N}(0, \sigma_{a,b}^2), \quad (5)$$

where $\mu_{a,b}$ is either $\mathcal{C}(a, b)$ as per Eq. (2) if modelling a sample-wise curve or $\mathcal{C}(a, n, b)$ for some (implicit and not further specified) training set size n as per Eq. (3) when modelling an iteration-wise curve. It assumes a noise that follows a Gaussian distribution with zero mean and dispersion $\sigma_{a,b}^2$ that may

vary over different anchor sizes. The assumption of a Gaussian noise is reasonable, because most loss functions form an average over sample-wise scores, which implies a Gaussian distribution through the Central Limit Theorem. For simplicity, we will not make a difference between the two types of learning curves, so that anchors are always denoted as budgets b (regardless of whether this refers to training set size or iterations, epochs, trees, etc.).

The task of forming a learning curve model for one or multiple learners is inherently one of supervised machine learning and requires the ability to generalise across anchors (and possibly even learners). In practice, observations are only available for a finite number of learner-anchor combinations $O = \{(a_1, b_1), (a_2, b_2), (a_3, b_3), \dots\}$, and the task is to learn, for each learner a , a model $\hat{f}_a(b)$ that expresses the belief about $f(a, b)$ for *any* budget b , not only those in O . Therefore, it is not enough to simply create an explicit estimate of μ_{a,b_i} for the anchors $(a, b_i) \in O$, but some general *pattern* must be learned. Observe that a here is not a parameter of $\hat{f}_a(b)$ since one often does not generalise across learners. However, some approaches advocate a single model $\hat{f}(a, b)$ that estimates $\mu_{a,b}$ for any learner-budget *combination*, where $a \in A$ and A is the set of all possible learners (Klein et al., 2017a,b; Swersky et al., 2014).

During this process of building a learning curve model, one generally needs to cope with two types of uncertainty. First, the *aleatoric* uncertainty is $\sigma_{a,b}^2$, which is intrinsic and averaged out in the true learning curve. Again, this is the uncertainty arising from randomness in the learner itself (if applicable) and random effects in the splits (or more general: data collection) when computing the empirical learning curve. Second, the *epistemic* uncertainty is the one the learning curve model *itself* has about the estimate of the mean value $\mu_{a,b}$. This uncertainty can be removed by gathering more observations (i.e., extending observation set O).

It is important to understand that epistemic uncertainty generally does not indicate model quality. Epistemic uncertainty is not related to correctness: A model can have no epistemic uncertainty (be absolutely sure) about an actually wrong prediction, and similarly, it can be uncertain about a prediction that is actually correct. Also, epistemic uncertainty gives no indication about whether the *class* from which the predictive model is inferred is suitable for the task, i.e., whether the true curve can be captured by the model that is fitted (e.g., a power law). Hüllermeier and Waegeman (2021) discuss this for the more general case of selecting an appropriate machine learning model. Recent results suggest that many learning curves are not adequately captured even by a very flexible parametric model, i.e., the 4-parameter MMF model (Kielhöfer et al., 2024), which motivates other, possibly non-parametric approaches for modelling, such as the one used in freeze-thaw Bayesian optimisation (Swersky et al., 2014). Therefore, the uncertainty at the meta-level about whether a model *class* is suitable for a task cannot be captured in epistemic uncertainty and must be studied independently.

Modelling uncertainty in learning curve models typically implies modelling the *epistemic uncertainty* about the curve mean $\mu_{a,b}$. This uncertainty refers to *arbitrary* anchors b , both those from the observed data O as well as the anchors that were not part of this. The three common patterns to define beliefs about $\mu_{a,b}$ are (i) *point estimates* (no uncertainty is expressed), (ii) *range estimates*, e.g., confidence intervals for $\mu_{a,b}$, or (iii) *distribution estimates*, which quantify a full belief model over the true value of $\mu_{a,b}$. The aleatoric uncertainty can also be quantified and modelled, e.g., by taking many different samples per anchor.

Fig. 8 illustrates all of these concepts together. The blue line $\mu_{a,\cdot}$ represents the true learning curve as per Eq. 2 or Eq. 3 for all possible values of b . As this integrates out all possible data splits as well as random factors from the algorithm, each point is an average from a distribution of many possible performance values. The variance of this distribution is expressed as $\sigma_{a,\cdot}^2$. This variance stems from the aleatoric uncertainty, in the sense that it is high when the aleatoric uncertainty is high, and vice versa. As the budget b (represented on the x-axis) increases, the aleatoric uncertainty typically decreases, and thereby the variance naturally decreases as well. The orange elements are related to observations of the learners' performance and the learning curve model. The orange points (which all together form O) show observations made on the empirical learning curve (as a sample from the blue distributions, this can be one or more per point). Note that these do not necessarily need to align with $\mu_{a,\cdot}$ or fall within the variance bandwidth. The orange solid line shows a point estimate defined by a parametric model obtained from the observations, and in this example, it substantially deviates from the true curve (cf. Sec. 3.1). The orange shaded area is a range estimate modelling epistemic uncertainty, which grows as one moves away from the available data (cf. Sec. 3.2). Finally, the dashed orange lines are distribution estimates for different budgets b (cf. Sec. 3.3). Each of these dashed lines can be seen as a probability density function (rotated 90 degrees) for a certain budget b . Each curve shown sketches the distribution of the belief about where the true mean may be situated, in this figure modelled through Gaussian distributions. As we move away from the observations, the shape of these bells grows bigger, indicating higher epistemic uncertainty.

In the following sections, we will explain the three model types in more depth. We explain the concept for models of the type $\hat{f}_a(b)$, i.e., models for a specific learner a , because this is the most common case. Generalising a curve model across learners requires additional logic, which we discuss along with the models that utilise such generalised model in Sec. 5.2.3.

3.1 Point Estimates of the Learning Curve

The simplest type of learning curve model for a learner a just estimates the mean curve values $\mu_{a,b}$ for any possible budget b and ignores uncertainty aspects (solid orange line in Fig. 8). Given an empirical learning curve in the form of some finite samples from this process at different anchors $B =$

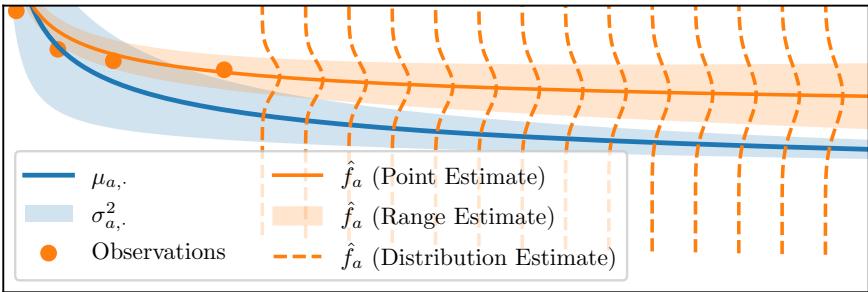


Fig. 8: Visualisation of various forms of uncertainty in learning curve modelling. The blue line $\mu_{a,.}$ represents the true learning curve; the dot · means that it is for all budgets b . The variance over all possible curves that could be sampled is expressed as $\sigma_{a,.}^2$. The variance stems from the aleatoric uncertainty. Orange points are anchors at which a learner's performance has been observed. From this, an empirical learning curve can be modelled. The orange solid line, area, and dashed lines are point estimate, range estimate, and distribution estimate models, respectively. The latter two models also express epistemic uncertainty.

$\{b_1, \dots, b_n\}$, a regression model $\hat{f}_a(\cdot|\theta) : \mathbb{N} \rightarrow \mathbb{R}$ is trained with respect to some model class with parameters θ . A considerable number of different parametric models have been proposed over time for this task. To our knowledge, the first proposal of such classes was made, apparently independently, by [Cortes et al \(1993\)](#) and [John and Langley \(1996\)](#) with the three-parametric inverse power law (IPL)

$$\mu_{a,b} = \alpha + \beta b^{-\gamma}, \quad (6)$$

where the parameters $\alpha, \beta, \gamma > 0$ need to be optimised to fit the learning curve for learner a . [Frey and Fisher \(1999\)](#) took a simplified variant of that model ($\alpha b^{-\beta}$) and compared it to a logarithmic ($\alpha \log b + \beta$), and an exponential model ($\alpha \cdot 10^{-\beta b}$). While it has been argued, at least for the power-law family, that there is a theoretical foundation for it ([Seung et al, 1992](#)), the considered model classes are typically not theoretically motivated but rather pop up in an ad-hoc manner. For example, [Gu et al \(2001\)](#) extended the above three classes, without a specific motivation, by a vapor pressure model, the Morgan-Mercer-Flodin (MMF) model, and a Weibull model.

Depending on the purpose of the model, it is essential to distinguish between best-fitting and best-predictive models. As was pointed out by [Gu et al \(2001\)](#), the model class that can best accommodate a given set of anchor points is not always the one that will make the best predictions on a high anchor when having been fit only on some initial anchors. To understand the learning curve of a learner on a given dataset, one is interested in a best-fitting model class. For extrapolation, one is interested in a best-predictive one.

Our work does not seek to give a broad overview of different model classes but rather about the usage of such models. We expose the inverse power law model because it is arguably the most prominent model class and has been advocated by many authors as a good fit for nearest neighbours, SVMs, decision trees, and neural networks (Frey and Fisher, 1999; Gu et al., 2001; Hess and Wei, 2010; Richter and Khoshgoftaar, 2019). However, other models have been proposed, e.g., based on differential equations (Boonyanunta and Zeephongsekul, 2004) or other physical laws (Gu et al., 2001), and authors have argued that other models, such as logarithmic shape can be a better fit (Gu et al., 2001; Singh, 2005). For an updated and exhaustive overview of used model classes, we refer to the work by Viering and Loog (2023).

3.2 Range Estimates of the Learning Curves

It has been recognised that incorporating some notion of uncertainty into the model itself is important (Mukherjee et al., 2003). Formally, this amounts to learn a *range estimate* function $\hat{f}_a : \mathbb{N} \rightarrow \mathbb{R}^2$ such that $\hat{f}_a(b) \equiv [u, v]$ with some pre-defined semantic relationship between $\mu_{a,b}$ and the interval $[u, v]$. In Fig. 8, this type of estimate is visualised through the orange area.

The semantics of the interval $[u, v]$ depend on what exactly is being modelled, which also implies *how* the model is created. In the earliest known attempt on this matter, Mukherjee et al (2003) model the (believed) *interquartile range* of the actual distribution $f(a, b)$ with this interval, i.e., values that are expected to be observed with a certain probability if sampling at a specific anchor (aleatoric uncertainty). In this approach, two parametric (i.e., inverse power law) models are built, one from the 25 and one from the 75 quantile for each observed anchor b . Even though the mean does not necessarily lie between these quartiles in general, this is the case in a Gaussian distribution, which is a sensible assumption as explained above. In contrast, Figueroa et al (2012) use it to model a confidence interval of $\mu_{a,b}$ (epistemic uncertainty). In this specific case, only one parametric model is learned, and the confidence interval around the curve is obtained through analytical rather than stochastic techniques. The confidence interval-based approach can also be thought of as putting a probabilistic bound on the gap between the predicted performance $\hat{f}_a(b)$ and the true value $\mu_{a,b}$.

Again, the mere presence of interval-based predictions that express epistemic uncertainty should not lead to the conclusion or belief that there is a necessary relationship to correctness. In particular, an epistemic uncertainty of 0 does not imply a correct prediction. Suppose the model \hat{f}_a is chosen from a class of which the mean curve $\mu_{a,\cdot}$ is not a member. In that case, it is *guaranteed* that there will be wrong predictions, regardless of the epistemic uncertainty expressed by the model. But even if $\mu_{a,\cdot}$ is among the models from which \hat{f}_a can be built, it can still (and usually will) happen that, based on insufficient observations, a wrong \hat{f}_a will be picked. In such a case, it is still conceivable that, depending on the probabilistic model on which \hat{f}_a rests, the epistemic uncertainty would be 0 for anchors b whereas $f(a, b) \neq \hat{f}_a(b)$; here $\hat{f}_a(b)$ is just

a value since the epistemic uncertainty of 0 means that the interval $\hat{f}_a(b)$ has only one value.

3.3 Distribution Estimates of Learning Curves

In a more ambitious case, we can try to learn a full *belief model* of the learning curve \mathcal{C} . Formally, this amounts to learn a *distribution estimate* function $\hat{f}_a : \mathbb{N} \rightarrow \{p \mid p \text{ is a distribution in the domain of the performance measure}\}$. Fig. 8, this model corresponds to the sequence of dashed orange distributions (in this figure, displayed for only 14 values of budget b).

As with the previous approaches, one typically uses parametric functions (e.g., the inverse power law) as a basis but specifies *distributions over their parameters* instead of a single (based on the maximum likelihood) assignment. The distribution over parameters then induces a distribution of the space of learning curves. Such a belief model is, for example, well-defined in a Bayesian framework that defines the posterior distribution of models given the observed data and assumes a certain model class. This posterior distribution cannot be efficiently computed exactly but approximate it through sampling (Domhan et al, 2015; Klein et al, 2017b).

Clearly, distribution estimates are the most flexible way of modelling uncertainty and allow many interesting operations. In particular, one can quantify the probability that the limit performance of a learning curve will be above or below some threshold τ , which is very useful for confidence-based early discarding (Domhan et al, 2015).

4 A Framework to Categorise Learning Curves Methods for Decision Making

Based on the common ground of learning curves and their models introduced in Sec. 2 and Sec. 3, this section presents a framework for *categorising* decision-making methods that use learning curves. We identify three orthogonal criteria along which those approaches can be categorised. The first criterion relates to the *decision-making situation* in which learning curves are used. We discuss these situations exhaustively in Sec. 4.1. The second dimension covers the *technical question* that is answered about a learning curve to support the decision. For example, are we interested in the saturation point or a complete model? These technical questions are sketched in Sec. 4.2, and we structure the literature review of Sec. 5 according to this axis. Finally, different *data resources* can be used to conduct an analysis with learning curves, e.g., other learning curves or features describing the datasets or the learning algorithms. These resources are covered in Sec. 4.3.

4.1 Types of Decision-making Situations

Learning curves are an important resource in at least three types of decision-making situations:

1. *Quantitative Data Acquisition* Consider the situation where a data scientist has a model trained on a set of observations, the performance is known, and there is the option to spend additional resources (e.g., money or labelling effort) to obtain additional training observations. The decision that needs to be made is: The acquisition of how many more labels is economically reasonable? This question has an obvious connection to the field of *active learning*, which addresses the question of *which* instances should be labelled next (qualitative acquisition). Another question is whether we should acquire other features instead.
2. *Early Stopping* (of training an independently considered model). In the situation where one is committed to some specific learner (a learning algorithm *and* its hyperparameters), minimising the training effort is a reasonable goal. Specifically, if large amounts of data are available and training is costly, the aim is to train until the saturation point is reached. Being able to detect or predict whether a learner's performance saturates after a given number of observations or iterations can support making decisions on this.
3. *Early Discarding* (in model selection). Similarly, if we want to *select* from various models, we want to stop the evaluation of a candidate when we are sure that it is not *competitive* to the current best solution. We compare the learner performance to that of *another* learner instead of its own performance on more training investment. For example, consider the situation where the learning curve of an algorithm seems to approach the saturation point, and we have already seen a superior model before, of which it is unlikely that the current algorithm will improve over. In this case, we can discard the performance of this learner based on the performance in relation to other models.

There is a large methodological overlap in creating a decision basis among all these decision-making situations. For example, whether more data points would be helpful to improve performance is related to the question of the training size that should be chosen to minimise training effort. Both questions, at their core, ask for the saturation point of the learning curve.

In the following sections, we will discuss each of the three decision-making situations in more depth.

4.1.1 Quantitative Data Acquisition

Quantitative data acquisition focuses on the question of how *many* training examples should be considered, given that they are all sampled i.i.d. from the same source. Quantitative data acquisition does not consider or pay attention to the possibility of acquiring *specific* instances, which would be considered in qualitative data acquisition. Qualitative data acquisition is mainly studied in the field of *active learning* and does not ask whether or how many instances should be acquired but for *which* instances a label should be acquired. Since active learning undermines the i.i.d. assumption, it generates a different type of learning curve and is not covered in this survey.

The relevance of learning curves for quantitative data acquisition rises from their ability to give insights into intrinsic properties of the data *source* as well as into the relationship between the number of training examples and the *utility* of having that number of samples. On the one hand, intrinsic properties refer to the intrinsic noise of the data (Cortes et al, 1994), which tells us about the best possible performance of any learner no matter how much data would be available from the source. If we know that, with the given data, we already achieve a performance close to the intrinsic noise, then data acquisition should focus on acquiring additional *features* instead of new instances. On the other hand, the utility is mainly determined by the cost of acquiring (additional) examples, the performance obtained with a certain number of samples, and the cost to train a model with a given number of instances (Last, 2007, 2009; Weiss and Tian, 2008). In this economic context, there are mainly five questions that can be considered:

1. *Possibility.* Can the classification performance be improved by more data?
2. *Potential.* What is the best possible predictive performance given unlimited training observations?
3. *Maximization Principle.* By how much can the predictive performance be improved if there is a budget for a fixed number of additional data points?
4. *Minimization Principle.* How many instances are necessary to obtain a certain degree of predictive performance?
5. *Utility maximization.* Which sample size maximises a given utility function?

In the context of data acquisition, we typically deal with sample-wise curves. Furthermore, one is often not committed to a particular learner; therefore, the performance measure in the above questions is implicitly the best one of a *portfolio* of learners. That is, one assumes a set of learners that is considered admissible for the prediction task due to external restrictions. In general, due to the ability to parameterise learners, this set is usually infinite. When referring to the portfolio's performance at a specific anchor point, we are interested in its best-performing algorithm at that specific point at the learning curve (Mohr and van Rijn, 2023). Of course, if one is committed to one particular learner, then the situation simplifies to a portfolio of size 1.

Since data acquisition is not for free, it is sensible to relate potential predictive performance improvements with the costs to collect the additional labels. Therefore, instead of looking only at predictive performance, one looks at *utility* of an anchor point. While performance typically only improves with an increased number of observations, the utility also considers acquisition costs, which negatively affect the utility. Therefore, the goal is to decide how many instances should be labelled to maximise utility, i.e., how many additional instances are justified before the added value no longer outweighs the additional costs (Last, 2007; Weiss and Tian, 2008).

4.1.2 Early Stopping

Early stopping means interrupting the training process of a learner if the learning curve has converged. The term ‘early’ refers to the fact that the learning process would normally be continued, e.g., because more data is available or other stopping criteria are not yet satisfied. That is, one uses the learning curve to judge that, despite more available resources or other criteria that would encourage further training, investing more time will not improve the performance of the considered model class any further. The training process can then be stopped early, i.e., earlier than if that criterion would not be used. Fig. 9 shows this logic in the left (red) part in which the blue learning curve of learner *a* is used to detect that not all the data is necessary and only 500 training instances are used (to save training time).

Early stopping can be applied to both sample-wise curves and iteration-wise curves. Early stopping in sample-wise curves means retraining a model on different training set sizes to create an empirical learning curve with training set size as the budget. This can make sense if we do not already know that the saturation point is larger than the available dataset size; otherwise we should immediately train on the complete dataset. We can then try to analyse the sample-wise curve of the learner for increasing training sizes and stop as soon as we find that performances do not change significantly between two anchors (John and Langley, 1996; Provost et al, 1999). For iterative learners (such as neural networks), an iteration-wise curve is usually a by-product that can cheaply be created in parallel to learning. Hence it might seem more appropriate to do early stopping based on an iteration-wise curve rather than the sample-wise curve. An additional advantage of early stopping in iteration-wise curves is that it can help avoid over-fitting, e.g., in neural networks (Bishop, 1995; Goodfellow et al, 2016) or gradient boosting. While it is conceivable that building sample-wise curves, even for iterative learners, could be useful in some cases, we are not aware of any such work being done for early stopping (or any other purpose).

The early stopping problem can be addressed retrospectively and projectively. Retrospective early stopping means to stop after observing the saturation point. Projective early stopping means to *predict* the saturation point before it is reached and stop precisely at the (believed) saturation point. The projective approach is particularly important in the case of sample-wise curves.

Early stopping in sample-wise curves and data acquisition might seem similar since both define a sample size at which a process should be stopped. However, the concepts are fundamentally different in three ways:

1. *Stopped Process*: Early stopping means to stop a *training* process (at the saturation point) run in a *machine*. In contrast, the decision-making situation in data acquisition is to stop the *data acquisition* process (at the economic saturation point). The latter is sometimes carried out by *humans*.

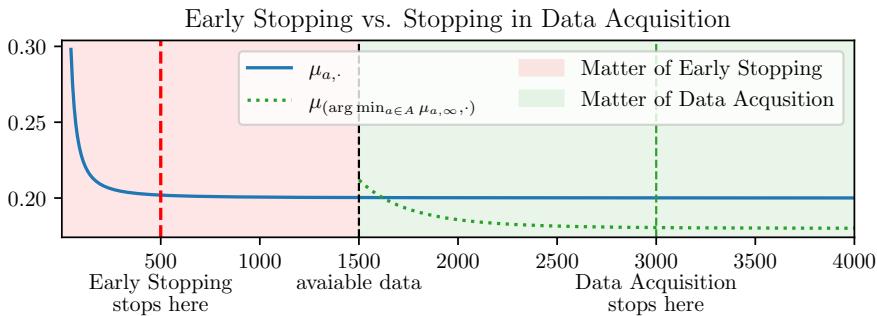


Fig. 9: Early stopping with sample-wise curves stops the training process of a single learner a at its saturation point using its (actually empirical) learning curve. It shows the learning curve of two learners, the blue curve and the green dashed curve. The latter has the best saturation performance. The available data marks a restriction to this process. In contrast, data acquisition considers the available data as the decision variable, and it stops collecting data when *all* learners have reached their saturation performance.

2. *Role of Available Data:* In early stopping, the available amount of training data is a given constraint under which early stopping operates, and data acquisition precisely seeks to control this quantity in an economically optimal fashion.
3. *Used Performance Curve:* Early Stopping uses a single learning curve of learner a to decide upon early stopping of the training of a . Data acquisition uses the learning curves of all learners under consideration (i.e., a finite set A) and considers the budget-wise best performance achievable (by *any* learner).

To clarify this difference, consider also the right (green) part of Fig. 9. The learner a may be the best solution available given the 1500 data points (in this case, a barely needs 500 of them to attain saturation performance). However, if more data *were* available, then at least one other learner could take advantage of that additional data and outperform a . The figure shows the curve of an optimal learner $a^* \in \arg \min_{a \in A} \mu_{a,\infty}$ that has the best performance if no limit is posed on the available training data (as in Cortes et al (1994)). Only after 3000 instances, *no* learner will improve the overall possible performance anymore; therefore, at this point, the data acquisition process stops.

4.1.3 Early Discarding

In many setups, the learner itself is a matter of optimisation. Consider the situation where we have a (possibly infinite) *set* of learners, e.g., a finite set of algorithms, each of which can be instantiated with a possibly infinite number of hyper-parametrisations. The task is to find the (hyper-parametrised) learner which performs best for the given data of size n in the sense that it creates,

on average, the best model. Formally, if A is the (infinite) set of parametrised learners, the goal is to find

$$\arg \min_{a \in A} \mathcal{C}(a, n). \quad (7)$$

This task is commonly known as *model selection*.

While it is uncommon in literature to be so explicit and describe the model selection problem through the value of the learning curve at some sample size, this formulation is rather precise and insightful. It emphasises that which learner is best might depend on the number of available training points. Note that one needs to separate some portion of the data for validation in practice to *estimate* model performances. In other words, most approaches in practice do not even address the above problem but instead

$$\arg \min_{a \in A} \mathcal{C}(a, \lceil \alpha n \rceil), \quad (8)$$

where $\alpha \in]0, 1[$ (open interval) is the training portion, typically between 70% and 90%, where the remaining portion of $1 - \alpha$ is used to estimate $\mathcal{C}(a, \lceil \alpha n \rceil)$, typically in some (possibly repeated) hold-out validation.

We could conceive that this procedure of estimating $\mathcal{C}(a, \lceil \alpha n \rceil)$ might involve the construction of an empirical learning curve as a sub-routine or on-the-fly. First, if a is an iterative learner, then the model performance $\mathcal{C}(a, \lceil \alpha n \rceil) = \mathcal{C}(a, \lceil \alpha n \rceil, t^*)$ is the performance of the iteration-wise curve at some point t^* where the learning process is stopped. Therefore, the whole iteration-wise curve $\mathcal{C}(a, \lceil \alpha n \rceil, t^*)$ is available for analysis. Second, even if a is not incremental, one could create a schedule $\{\alpha_1, \dots, \alpha_k\}$ of increasing $\alpha_i \leq \alpha$ and thereby create a sample-wise curve (Mohr and van Rijn, 2023). Such a sample-wise curve would also offer the perspective, via extrapolation, to address Eq. (7) rather than just Eq. (8).

In the light of the availability of such a (partial) empirical learning curve, *early discarding* is the practice of aborting the performance estimation procedure of a candidate as soon as it becomes apparent from that curve that the candidate cannot be the solution to the above optimisation problem.

Formally, this is to drop a candidate a as soon as the criterion

$$\mathcal{C}(a, b_{ref}) > \min_{a^* \in A} \mathcal{C}(a^*, b_{ref}) \quad (9)$$

can be verified, where b_{ref} is usually n or $\lceil \alpha n \rceil$. In other words, as soon as it can be shown that a is not the best learner of all possible learners A , no further resources should be committed to training learner a . Note that, even though the terms are frequently mixed up in literature, this is very different from early stopping, in which the convergence of the curve of a single learner is considered in isolation (cf. Fig. 1).

Early discarding has been applied to both observation (Adriaensen et al., 2023; Mohr and van Rijn, 2021, 2023; Ruhkopf et al., 2023) and iteration (Adriaensen et al., 2023; Domhan et al., 2015; Klein et al., 2017b; Ruhkopf et al., 2023;

[Swersky et al, 2014](#)) learning curves. In the first case, one is sampling from $\mathcal{C}(a, \cdot)$ at different anchors n and hopes to be able to drop sub-optimal candidates at $n \ll b_{ref}$ anchors (much) smaller than the target size b_{ref} . In the second case, one always uses the complete dataset (or at least all data designated for training, say n), observes samples of $\mathcal{C}(a, n, \cdot)$ for different anchors (maybe epochs) t , and seeks to avoid convergence if it can be foreseen that the convergence performance will be sub-optimal.

Early discarding is more aggressive than early stopping because it does not need to wait until the learning curve converges. On the contrary, one tries to avoid reaching convergence since this is considered a waste of resources in the case that the learner performs sub-optimal. In an extreme case and depending on available knowledge about learners (cf. Sec. 4.3), one could only use a single point of an empirical learning curve to discard a candidate.

Situations in which early discarding plays a role can be further classified into horizontal and vertical scenarios (and a mixture of the two):

1. *Horizontal Model Selection.* Horizontal model selection implies an apriori fixed *finite* set of learning algorithms, from which one has to be selected. Empirical learning curves are grown iteratively for the whole set or shrinking subsets of it. Successive halving and related works are a prominent example of horizontal decision making ([Van den Bosch, 2004](#); [Jamieson and Talwalkar, 2016](#)). However, these approaches only consider the last anchor point (rather than the complete learning curve).
2. *Vertical Model Selection.* Vertical means that the set of learners is generally not limited to a finite set, and the set of *evaluated* learner candidates evolves over time (i.e., not fixed apriori). Learners are evaluated one *after* another. Each learner is evaluated in an iterative fashion to grow a learning curve and allow for early discarding. Examples are the early discarding routine for deep networks by [Domhan et al \(2015\)](#) or, more generally, for learning curve cross-validation ([Mohr and van Rijn, 2021, 2023](#)).
3. *Diagonal Model Selection.* This case is similar to the vertical decision-making situation with the difference that one does allow to *continue* the evaluation of a candidate at a later point. Hence, candidates are not evaluated one after another, but the evaluation of different candidates can be interleaved. Examples are Bayesian optimisation-based approaches to pause and continue evaluations of (not necessarily iterative) learners ([Klein et al, 2017a](#); [Swersky et al, 2014](#)). Non-iterative learners must be trained from scratch with the increased budget.

Another approach that addresses this type of decision-making situation is Hyperband ([Li et al, 2017](#)) and Bayesian optimisation based on progressive sampling ([Zeng and Luo, 2017](#)). However, neither of these approaches considers learning curves even though they implicitly construct them. Decisions are taken based on the observations of the largest anchor point considered so far.

4.2 Technical Questions Asked About Learning Curves

A plethora of questions can be asked about learning curves. Fig. 10 gives an overview of these questions. The figure is organised in three layers (depth dimension) corresponding to the three types of estimates discussed in Sec. 3. Each layer consists of a set of questions that can be posed about learning curves. From bottom to top, the questions are ordered by complexity, and an arrow from one question to another indicates that the question with the incoming arrow is more general. Answering the more general question also implies answering the less general question.

In the simplest case, we can answer a binary question. There are four relevant questions, i.e., (i) whether some specific anchor point, e.g., the dataset size, is beyond the saturation point ($b_{sat} \leq b_{ref}$), (ii) whether the performance of a learner at the saturation point is better than some baseline τ ($p_{sat} \leq \tau$), (iii) whether the performance $p_{ref} := \mathcal{C}(a, b_{ref})$ at some reference point b_{ref} is better than some threshold τ , or (iv) whether a specific anchor point is beyond the *utility-based* stopping point ($b_{sat}^u < b_{ref}$). To our knowledge, the only approaches in this category are those implicitly answering question (iii) by discarding candidates that are not believed to be competitive (see, e.g., [Jamieson and Talwalkar, 2016](#); [Petrak, 2000](#); [Zeng and Luo, 2017](#)); here $\tau = \min_{a \in A} \mathcal{C}(a, b_{ref})$ is the (unknown) best performance of any learner on the target size.

A family of slightly more general questions tries to *order* a set A of learning algorithms w.r.t. their performance at some (future) anchor b_{ref} . We denote this ordering as $\pi_{a \in A} \sim \mathcal{C}(a, b_{ref})$. Here, b_{ref} is typically the maximum available training data, even if an iteration-wise curve is considered because then this is the termination performance of that curve. In the simplest case, we could ask for a concrete pair of two learning algorithms a_1, a_2 whether $\mathcal{C}(a_1, b_{ref}) \geq \mathcal{C}(a_2, b_{ref})$, i.e., which will perform better at some reference point. For example, the work by [Leite and Brazdil \(2005\)](#) answers this question. If this question is simultaneously asked for a set of or even all possible pairs of algorithms, one asks for a partial or even the full ranking $\pi_{a \in A} \sim \mathcal{C}(a, b_{ref})$ of algorithms. A particular case is to ask only for the best algorithm $a^* = \arg \min_a \mathcal{C}(a, b_{ref})$, which implicitly answers that $\mathcal{C}(a^*, b_{ref}) \leq \mathcal{C}(a, b_{ref})$ for any learner a but without explicitly asking for any other comparisons. Still, the comparison is merely qualitative, and answering this question does not necessarily require to quantify any aspect of any learning curve.

The above questions are merely qualitative and not quantitative, which gives rise to a third level of complexity, where the concrete values of b_{sat} ([Provost et al, 1999](#)), p_{sat} ([Cortes et al, 1993](#)), p_{ref} ([Baker et al, 2018](#); [Chandrashekaran and Lane, 2017](#); [Leite and Brazdil, 2003, 2004](#)), or b_{sat}^u ([Weiss and Tian, 2008](#)) are being modelled. Here, the reference performance p_{ref} is the performance at a fixed reference point, often the number of training samples available for cross-validation.

All questions up to this point produce closed answers in the sense that the answer is either a boolean value, a number, or a finite ranking of candidates.

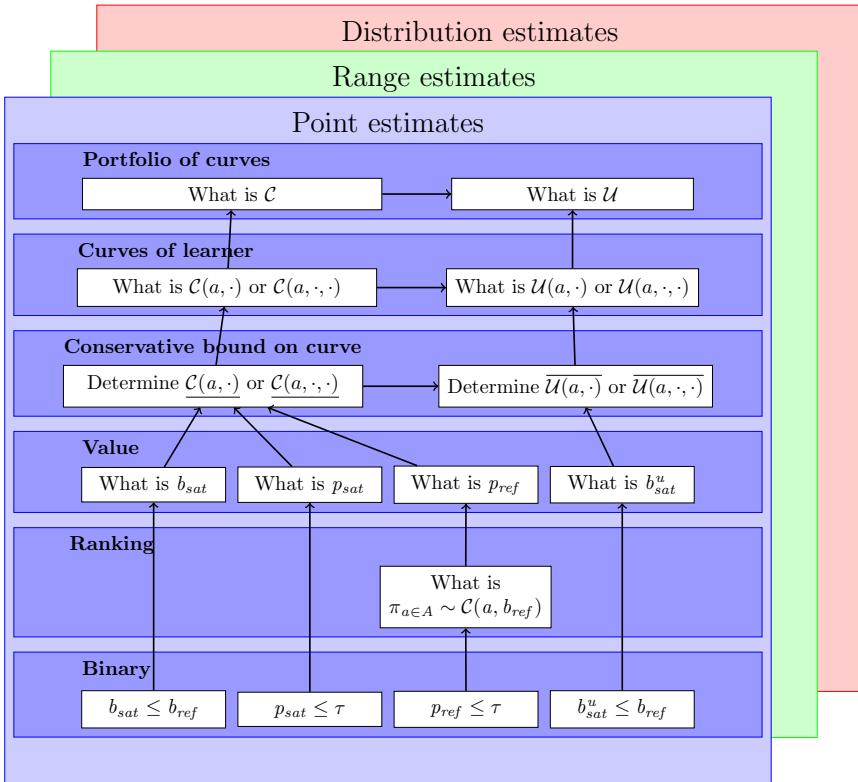


Fig. 10: Technical questions that can be asked on learning curves.

At the fourth level, the task is to make assertions about arbitrary points of a learning curve. However, the answers do not yet refer to the value of the learning curve itself but only a *bound* on those values.

At a fifth level, we could eventually ask for a model of the whole learning curve of a learner a , i.e., $\mathcal{C}(a, \cdot)$ for sample-wise curves (Figueroa et al, 2012; Frey and Fisher, 1999; Gu et al, 2001; John and Langley, 1996; Mukherjee et al, 2003) or $\mathcal{C}(a, n, \cdot)$ for iteration-wise curves given a fixed (sample) anchor n (Cortes et al, 1993; Domhan et al, 2015). We can ask a similar question for the utility curve. While the question is on the same level, it is more general, as it is *based* on the learning curve itself (Last, 2007, 2009; Weiss and Tian, 2006) and combines it with other information such as acquisition costs.

Finally, at the sixth and most general level, we could ask for a model of the whole performance function \mathcal{C} , i.e., the model of the learning curves across all learners (Klein et al, 2017a,b; Swersky et al, 2014). Analogously, this question could be asked for the whole utility function \mathcal{U} , which is arguably the most complex and general question that can be asked, although we are not aware of any works that have done so.

Since all of the above questions are answered based on observational statistics, we can also consider noise and uncertainty aspects for the quantitative questions. In the simplest case (blue layer, cf. Sec. 3.1), we only get point estimates, i.e., the estimate of \mathcal{C} at one or a set of points. Since these estimates are always afflicted with uncertainty, it is reasonable to ask for quantifications of this uncertainty. One form is to express *strict bounds* as in the fourth layer, which may be derived from assumptions about the learning curve shape, e.g., convexity (Mohr and van Rijn, 2021, 2023). Another form is to express *probabilistic bounds* like confidence intervals around \mathcal{C} (Figueroa et al, 2012; Koshute et al, 2021) (cf. Sec. 3.2). In the most general form, we can ask for a full *belief model* of the learning curves, specifying a probability distribution over the values of \mathcal{C} at one point or a set thereof (Domhan et al, 2015; Klein et al, 2017a,b; Swersky et al, 2014) (cf. Sec. 3.3).

It is common practice to solve relatively simple questions by implicitly answering more complex ones. For example, a typical question in the context of model selection is whether the performance of a candidate learner at some given data or in the limit will beat a known baseline (Domhan et al, 2015; Leite and Brazdil, 2005; van Rijn et al, 2015). This is the binary question of $p_{\text{ref}} \leq p^*$, where p^* is the best-known performance. Often, this question is answered by *estimating* p_{ref} (maybe p_{lim}) explicitly and then comparing it to p^* (Leite and Brazdil, 2005; van Rijn et al, 2015), which is an answer to a slightly more complicated question. Domhan et al (2015); Swersky et al (2014) built an explicit curve model, for estimating p_{lim} and p_{ref} , respectively. The approach answers this binary question by building an entire learning curve model and then derives the binary answer from it. The rationale behind this is the notion that one often needs rather complex models to find a high-quality answer to a simple question, and it is just a side effect that one can then even answer other questions with those models.

4.3 Used Data Resources for Inference

Above we have discussed a series of questions that can be asked about learning curve properties, which in turn are important for decision making in a specific context. Of course, answering these questions requires specific informative resources. In a concrete decision-making situation, we are typically confronted with a dataset and a learner or a portfolio of learners. We call this the *target* dataset and the *current* learner. That is, we want to say something about the learning curve of the current learner in a domain in which we have a finite (the target) dataset d available.

Fig. 11 shows the types of data resources that can be used to answer questions about learning curves. We can utilise empirical learning curves gathered on the target dataset, empirical learning curves gathered on other datasets, dataset meta-features, and features describing the learners. Learning curves gathered on the target dataset come at a particular computational cost, as they need to be generated during the process. Typically, when modelling the current learner on the target dataset, a *partial* empirical learning curve is constructed,

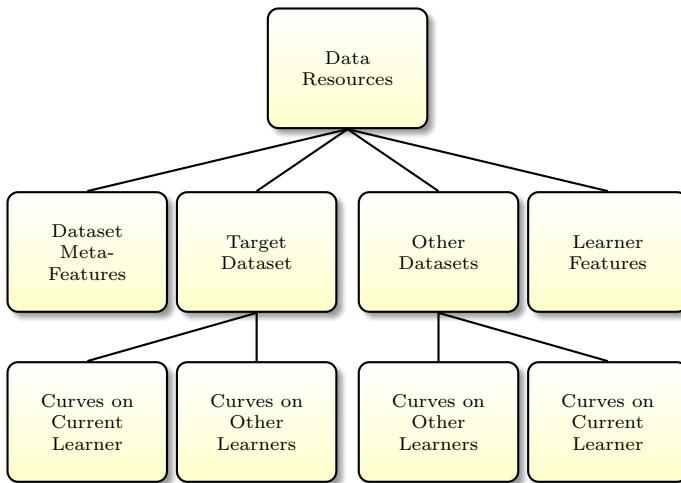


Fig. 11: Taxonomy of data resources for learning curve analysis

which can then be step-wise extended or discarded (Domhan et al, 2015; Leite and Brazdil, 2003; Provost et al, 1999). When using learning curves of other datasets, those are usually available up to a large portion of the dataset size (with a specific schedule of anchors) in the case of sample-wise curves or until convergence in the case of iteration-wise curves. This is because these curves could be prepared offline before the target dataset became available (Leite and Brazdil, 2003, 2010; van Rijn et al, 2015). Both learning curves of the current learner and other learners can be utilised for this. In the context of model selection, various learners are usually evaluated. Therefore we can acquire various learning curves of other learners on the target dataset (Baker et al, 2018; Chandrashekaran and Lane, 2017; Klein et al, 2017a,b; Swersky et al, 2014).

Other types of data resources that can be used are meta-features on the datasets and learner features. These are measurable qualities of the dataset and learner, respectively, and these can indicate how similar specific datasets (or learners) are. These give the decision-making algorithm a sense of which learning curves are more informative for the current learner and target dataset. To the best of our knowledge, the only line of research utilising meta-features for learning curve modelling is the work of Leite and Brazdil (2008, 2010); Ruhkopf et al (2023). The description of learners through features for the sake of model prediction is specifically prevalent in the analysis of iteration-wise curves (Baker et al, 2018; Klein et al, 2017b; Swersky et al, 2014). The development and analysis of meta-features is a research field in its own right; for more information, we refer the reader to Brazdil et al (2022).

5 Literature Review on learning curve extrapolation methods

This section presents the literature review that covers methods to model and utilise learning curves. We organise approaches based on the key problem they resolve on a rather abstract level and independent of the purpose or type of decision-making situation in which they were presented. We organise it along the framework presented in Fig. 10, particularly along the axis that reflects the technical question asked about a learning curve. The motivation to not use the type of decision-making situation, which is also a prominent property of these methods, is that the same approach can be used in *different* decision-making situations. For example, [Leite and Brazdil \(2004\)](#) present and motivate an approach to identify the portion of some given data that should be used for training (i.e., determine the saturation point), but the same approach could be used to determine how much more data would be needed to obtain saturation performance. Similarly, [Domhan et al \(2015\)](#) present an approach that aims to decide during training whether a neural network will become competitive (ask for saturation performance); however, they did this by modelling the complete learning curves.

Fig. 12 gives an overview of all the methods categorised in this framework. We make the following two observations.

- Most learning curve methods address the early discarding / model selection decision-making situation. This implies that there is an opportunity for more research on, for example, data acquisition or early stopping. We note that research towards active learning provides many approaches that handle data acquisition (which we do not cover), which might serve as a basis for a literature search.
- Second, most methods are centred around the middle levels of problem complexity they address, i.e., predicting the actual value of a learner at a certain point and predicting the complete curves of a learner. It seems logical that there are benefits for exploiting the situation where either the complete portfolio is modelled (e.g., the benefit of parameter sharing, the opportunity of model acquisition) or the binary problem is solved (because of the simplicity of the problem definition). Methods addressing the binary situation, such as Successive Halving and Hyperband, have attracted quite some attention.

To further structure the overview, we divide the whole literature on methods for learning curves into two roughly even groups according to the *usage* of a learning curve model as described in Sec. 3. Approaches in the first group do not employ a learning curve model. These approaches address the questions defined in the four lowest levels of the framework (see Sec. 5.1). In contrast, approaches of the second group, i.e., which employ a learning curve model, address questions in the top two levels of the framework (see Sec. 5.2). Approaches with a curve model are more general, but that does not necessarily mean they give better answers to simpler questions. In fact, [Kielhöfer](#)

	Data Acquisition	Early Stopping	Early Discarding / Model Selection	Unclear / Mixed
Portfolio			Swersky et al (2014); Wistuba and Pedapati (2019)	Klein et al (2017a,b)
Curves of Learner	Richter and Khoshgoftaar (2019)		Cortes et al (1993); John and Langley (1996); Gu et al (2001); Mukherjee et al (2003); Figueira et al (2012); Domhan et al (2015); Cardona-Escobar et al (2017); Adriansen et al (2023); Kielhöfer et al (2024); Egelé et al (2024)	Frey and Fisher (1999); Boonyamanta and Zeebhongsekul (2004); Singh (2005); Hess and Wei (2010); Kolachina et al (2012); Koshute et al (2021)
Conservative bound on curve			Sabharwal et al (2016); Mohr and van Rijn (2023)	
Value / Quantitative prediction	Weiss and Tian (2006, 2008)	Meek et al (2002); Bishop (1995); John and Langley (1996); Provost et al (1999); Leite and Brazdil (2003, 2004); Ng and Dash (2006)	Chandrashekaran and Lane (2017); Baker et al (2018)	Last (2007, 2009); Cortes et al (1994)
Ranking / Qualitative prediction			Leite and Brazdil (2005, 2007, 2008, 2010); van Rijn et al (2015); Ruhkopf et al (2023)	
Binary			Petrak (2000); Van den Bosch (2004); Jamieson and Talwalkar (2016); Li et al (2017); Zeng and Luo (2017)	

Fig. 12: Overview of the methods (indicated by the bibliographic reference) covered and categorised in this framework. Each method is categorised along the problem type they solve (vertical axis) and the type of decision-making situation they explicitly address (horizontal axis). Some methods are employed to address multiple decision-making situations. Some papers do not explicitly state which decision-making situation is being addressed, or address multiple. These are categorised in the last column. Several models can be used for more decision-making situations than the original paper evaluated them on. For example, the portfolio approaches could in theory be used for any decision-making situation.

et al (2024) show that the model-free MDS approach discussed in Sec. 5.1.2 in specific situations outperforms a parametric model such as the ones presented by Gu et al (2001) discussed in Sec. 5.2.1. In total, the approaches cover 10 of the questions discussed in Sec. 4.2, which are organized in the following sub-sections:

5.1. Approaches without learning curve model:

1. Is the target performance of a learner worse than the one of the best learner, i.e., $p_{ref} > \min_{a \in A} \mathcal{C}(a, b_{ref})$?
2. What is the ordering $\pi_{a \in A} \sim \mathcal{C}(a, b_{ref})$ of the learning algorithms w.r.t. their performance at some target anchor?
3. What is the saturation performance (p_{sat})?
4. What is the saturation point (b_{sat})?
5. What is the utility-based stopping point (b_{sat}^u)?

6. What is the value of the learning curve at a specific (fixed and known) point? ($\mathcal{C}(a, n)$ or $\mathcal{C}(a, n, t)$)
 7. What is a lower/upper bound of the learning curve at any point? ($\underline{\mathcal{C}}(a, \cdot)$ or $\overline{\mathcal{C}}(a, n, \cdot)$)
- 5.2. Approaches with a model for the learning curve:
1. What is the value of the learning curve at *any* (queryable) point? ($\mathcal{C}(a, \cdot)$ or $\mathcal{C}(a, n, \cdot)$)
 2. What is the *utility* at an arbitrary anchor point? ($\mathcal{U}(a, \cdot)$ or $\mathcal{U}(a, n, \cdot)$)
 3. What is the value of the learning curve at any queryable point of *any* queryable learner? ($\mathcal{C}(\cdot, \cdot)$)

We organise the rest of this section exactly according to this scheme.

For each approach, we always consider the most general problem it solves, independently of how this solution is used in the context of a paper. For example, Domhan et al (2015) decide whether the saturation performance of a learner beats some threshold (question at the binary level) but develop a learning curve model and are hence discussed alongside the approaches for question 1 in Sec. 5.2.1.

5.1 Approaches Without Learning Curve Models

Many interesting questions related to learning curves can be addressed without even building an explicit learning curve model. None of the questions in the lower layers in Fig. 10 necessarily requires a learning curve model. Fig. 13 shows a summary of all the approaches we are aware of, which make significant assertions or decisions related to learning curves without building a learning curve model.

5.1.1 Prediction of Candidate Competitiveness (Binary)

In this section, we discuss approaches that answer the early stopping criterion posed in Eq. 9 without using a learning curve model. The early discarding criterion can be seen as an instantiation of the binary question $p_{ref} \leq \tau$, where $\tau = \min_{a^* \in A} \mathcal{C}(a^*, b_{ref})$ is the best value that any learner on the available resources. This (conceptually simple) question is often answered by applying sophisticated learning curve models. The usage of learning curve models is understandable since, at the time of the decision, neither p_{ref} nor $\min_{a^* \in A} \mathcal{C}(a^*, b_{ref})$ is known; therefore, extrapolating learning curves offers a possibility to make assessments about these quantities. However, it is also possible to say something about the early discarding condition without a learning curve model.

As far as we know, the only type of approach in this category is *horizontal* early discarding with an *implicit* affirmation of the early discarding criterion that only uses the last anchor of the empirical learning curve. That is, during the model selection process, all remaining members of a candidate set A are trained to some budget. This budget can be either a sample size (sample-wise curves), learning iterations, or time (iteration-wise curves).

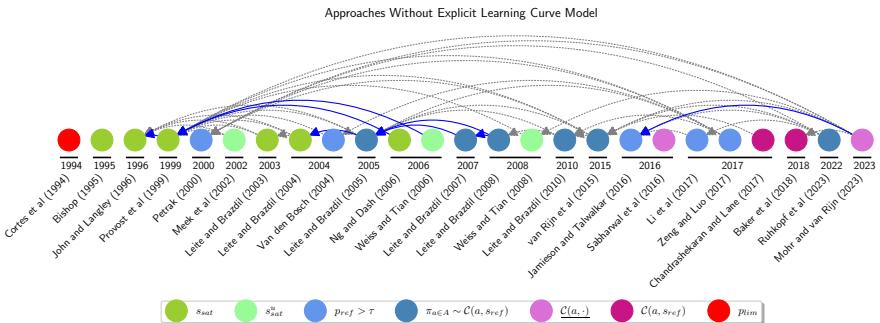


Fig. 13: Citations across approaches without learning curve models. The colours indicate the question addressed about learning curves. Normal arrows indicate that the paper cited the other paper, and coloured arrows indicate an experimental comparison with previous approaches.

Then, some portion of candidates A^- is removed from A based on the *last value* of their respective curve. The implicit assumption is that $\mathcal{C}(a, b_{ref}) > \min_{a^* \in A} \mathcal{C}(a^*, b_{ref})$ for every $a \in A^-$.

To our knowledge, the first algorithm in this line was introduced in the *Wrapped Progressive Sampling* procedure (WPS) by [Van den Bosch \(2004\)](#). In this approach, a dynamically computed subset of the candidate set A is discarded (instead of a constant fraction as 50%). WPS creates a histogram with ten bins b_1, \dots, b_{10} of the candidates in A based on their validation accuracy to decide which candidates are discarded. Then WPS identifies the largest index $i^- = \max\{i \in \mathbb{N}, < 10 : |b_i| < |b_{i+1}|\}$ of a bin that has fewer elements than its successor. Then, all candidates in bins with an index lower or equal i^- are dropped. In an extreme case, all except one candidate might be dropped right in the first iteration, e.g., if $|b_{10}| = 1$ and $|b_9| = 0$.

More recently, successive halving ([Jamieson and Talwalkar, 2016](#)) and hyperband ([Li et al, 2017](#)) have been introduced, which are both adequate methods based on a simple concept. Successive halving considers a set of candidate models A , which all receive an initial budget. It iteratively drops 50% of the candidate set A while doubling the remaining candidates' budget until a single candidate model remains. Hyperband is a series of successive halving brackets, where each bracket is initialised with an increased initial budget and a new initial set of candidates. [Zeng and Luo \(2017\)](#) proposed an extended version in which all the candidates that perform worse than the best candidate by some constant and pre-defined margin are discarded. Interestingly, the above approaches operate without a learning curve model for extrapolation and refrain from using the existing observations other than comparing the last seen values. Therefore, neither predictions nor observations of recent trends are being utilised. Nonetheless, these methods perform well empirically and come with theoretical guarantees.

5.1.2 Candidate Ranking

The methods in this section aim to rank the learning algorithms with respect to their (expected) performance at the full dataset size. In our scheme in Sec. 4.2 and Fig. 10, we denote this question as $\pi_{a \in A} \sim \mathcal{C}(a, b_{ref})$. Depending on the available resources (cf. Sec. 4.3), such a ranking can be based on the other learning curves available from other contexts and the explicit characterisation of such contexts, e.g., features that describe datasets or algorithms.

Ranking Without Context Description

The first approach we are aware of to address this problem was *metalearning on data samples* (MDS) presented by Leite and Brazdil (2005) through the notion of comparing two learners, i.e., a ranking of two. Given a dataset, MDS decides which of two algorithms is the better choice on a given dataset. Therefore, it can be used for early discarding. The authors specifically use an SVM and a C5 decision tree but rightfully claim that any algorithm could be used. The formal basis of the work is the same as the one introduced in their previous work (Leite and Brazdil, 2003, 2004). Similar to previous work, it assumes that empirical learning curves (with standardised anchor sizes) for the learner under examination are already known for *other* datasets. Additionally, it builds upon the idea of quantifying the *distance* between the target and the other datasets based on the sum of squared distances over the already-known performances at anchors of the target dataset. Once the most similar k learning curves have been identified, MDS assigns a score to each learner that is the *mean accuracy* of its k nearest neighbours (at the final anchor). It then selects the algorithm with the higher score.

Leite and Brazdil (2005) acknowledge that this method may result in poor rankings because even the closest learning curve on other datasets can still be substantially different and propose *learning curve adaption* as a remedy. Instead of forming the mean directly over the target anchors of the nearest neighbour empirical learning curves, the authors first *scale* those curves to make them more similar to the shape already observed on the target dataset. To this end, they compute a scaling constant under which the overall anchor-wise distance is minimised and then multiply all the scores with this constant. This version of the MDS algorithm is called AMDS (probably for Adaptive MDS).

Interestingly, one can argue that this adaption technique could be applied either *before* or *after* determining the k nearest neighbours. Doing it before could lead to other (and better) nearest neighbours because then the neighbours are determined more with respect to the *shape* of the learning curve, and the *offset* plays much less of a role. However, in the above paper, the adaption is done *after* retrieving the neighbours.

The authors extended the approach by creating an online sampling scheme with the SetGen algorithm (Leite and Brazdil, 2007). SetGen is an online adaption of AMDS in that, after each acquired anchor, it is decided whether and which anchor should be evaluated with each algorithm; this considers both

the (believed) accuracy and the runtime. This procedure can be seen as a way of racing between the algorithms. The potential of each additional amount of budget is judged based on the metalearning database.

An implicit assumption of all approaches in this line of research is that the datasets in the database from which performances are extracted need to be at least as big as the target dataset. This issue was first explicitly treated in the *Pairwise Curve Comparison* approach (PCC) (van Rijn et al, 2015). This algorithm builds upon the works of Leite and Brazdil (2010) and implements a voting scheme to identify the best learning algorithm of a portfolio; votes are distributed based on wins, which are determined based on the predicted performance at the complete dataset d . van Rijn et al (2015) explicitly discuss the issue if the sizes of datasets in the database and the target dataset are not identical. In general, the point for which predictions must be made is typically not one of the anchors; it is typically not a power of 2 but rather 90% of the given dataset size (due to the holdout scheme). A remedy is to resort to the closest available anchor in the schedule. However, if the highest anchor available for another dataset is much smaller than the required training size of the target dataset, then it is unclear how that curve should be used.

Ranking With Context Description

The first work we are aware of that realises an explicit context description was proposed by Leite and Brazdil (2008, 2010). Similar to the methods discussed earlier, these methods select the k nearest datasets to measure the relevance of known complete learning curves of other datasets for performance prediction on the target dataset. The main difference is that, in addition to the contribution of the partial learning curve itself to the distance, they also use the distance between the datasets in terms of their *meta-features*. More precisely, they compute the Manhattan distance between seven range-normalised dataset meta-features, e.g., dataset size, number of symbolic and numerical attributes, etc. The overall distance between the datasets is then the *sum* of the distance between the partial learning curves and the distance in terms of meta-features. This work was marginally refined in the *Selection of Algorithms using Metalearning* approach (SAM), which applies the same logic but assigns a *weight* to each of the two distance sources (Leite and Brazdil, 2010); the weight is however implicitly assumed to be set to 0.5.

A recent and entirely different approach to candidate ranking is the MASIF transformer framework (Ruhkopf et al, 2023). This approach takes partial learning curves of different learners on the current task, which may have potentially different lengths and combines them with dataset meta-features in order to predict *latent utility values* of each learner as expected for the complete dataset. The transformer is trained based on previous experiences on datasets for which true rankings among the learners have been computed for the complete dataset. It is unclear to which degree the utility values predicted by the transformer resemble the actual performance of the learners at the target

size, but the paper suggests that the *ordering* $\pi_{a \in A} \sim \mathcal{C}(a, b_{ref})$ of the learners according to these scores is relatively faithful to the true ordering induced by the actual performances.

5.1.3 Identification of Saturation Performance

Estimating the limit or saturation performance p_{sat} is helpful in two decision-making situations:

1. for sample-wise curves, it can be used for data acquisition by checking whether the availability of more data promises to improve predictive performance significantly.
2. for iteration-wise curves, it can be used for early discarding. Given a threshold p , we can determine whether training a model to convergence will achieve a generalisation performance of at least p .

To the best of our knowledge, the question on estimates of p_{sat} is the only of the above questions that received a substantial amount of theoretical contributions. This is not surprising since p_{sat} is an asymptotic quantity that is arguably suited for theoretical analysis. The root of this line of research is the statistical mechanics framework (Seung et al, 1992). This and related research (Amari and Murata, 1993; Fine and Mukherjee, 1999; Murata et al, 1992) consider a type of capacity curve in which asymptotic properties of the learning curve are expressed in terms of the number of parameters, usually those of a neural network. However, a side observation of these works is that there is a kind of symmetrical behaviour between the *train error* and the *validation error* (sometimes called generalisation error).

Cortes et al (1994) take these observations to use the mean of the two empirical curves to estimate p_{sat} as soon as the training error starts to rise, i.e., as soon as the model cannot accommodate the training data perfectly anymore. Although this work considers capacity curves to identify the intrinsic noise level of the data, i.e., the minimum error necessarily made by *any* learner, they also report the asymptotic performance of a neural network on a single data set.

To the best of our knowledge, this is the only approach that estimates p_{sat} without building an explicit learning curve model. While several other methods are capable of estimating p_{sat} of iteration-wise curves of neural networks (Domhan et al, 2015; Swersky et al, 2014), these rely on full learning curve models, which are therefore discussed in Sec. 5.2.1.

5.1.4 Identification of Saturation Point

Identifying the saturation point b_{sat} of sample-wise curves is useful in the following cases:

1. Early-Stopping with sample-wise curves: Which *portion* of the *available* data is necessary to obtain saturation performance?

This is relevant if $|d| > b_{sat}$ or the relationship between $|d|$ and b_{sat} is unknown and training on full d is potentially undesirable.

2. Early-Stopping with iteration-wise curves: How many iterations are necessary until performance converges? This applies to incremental learners, such as neural networks.
3. Data acquisition: How many additional labelled observations are necessary to obtain (near-optimal) performance? This applies if $|d| < b_{sat}$. The question can be posed for a specific learner or a portfolio.

Retrospective Approaches

The simplest way of determining the saturation point b_{sat} is to incrementally build a learning curve and stop as soon as it is believed that the saturation point has been *exceeded*. If we do this, we can estimate that the saturation point lies between the last two anchors. For iteration-wise curves, determining b_{sat} comes for free as a side-product of the training procedure. It is commonly used for training neural networks (Bishop, 1995; Goodfellow et al, 2016). On the other hand, it requires restarting and is potentially costly for sample-wise curves.

John and Langley (1996) define a *dynamic sampling* approach to determine the b_{sat} for sample-wise curves. A straightforward approach mentioned in that paper is to observe whether the performance has become *worse* on the last sampled anchor. If so, one might consider gathering empirical evidence that the saturation point has been *exceeded*, i.e., it should be somewhere between the last two anchors. However, the authors argue that preliminary results indicate that this approach often stops too early. This is mainly caused by high aleatoric uncertainty, which implies noisy empirical learning curves. On the other hand, one could argue that this approach stops far too *late* because it can require quite some iterations until, by chance, the observed performance is worse than the one of the last iteration. Therefore, John and Langley (1996) propose also a model-based approach to avoid this problem, which we discuss in Sec. 5.2.1. Of course, when having access to such a model, we can query the expected performance and compare it to the performance at the last anchor.

Provost et al (1999) address the stability issues and also (some of) the efficiency issues of the above trivial approach in a scheme they call *progressive sampling*. Similar to dynamic sampling (John and Langley, 1996), progressive sampling induces models for each anchor in the schedule until the convergence of the learning curve is detected. There are two main differences between the two approaches. First, the authors propose to use *geometrical* instead of arithmetic schedules, i.e., a schedule of the form b^k instead of bk , where b is a constant and k is the position of an anchor in the schedule. They prove that every geometric schedule is asymptotically optimal in terms of runtime; that is, every such schedule has the same asymptotic runtime as the schedule that evaluates only on b_{sat} . This optimality proof only holds in the asymptotic calculus; in practice, there are better and less good geometric schedules. For example, Provost et al (1999) propose a dynamic programming approach (called DP), which efficiently computes the cost-optimal schedule based on a prior distribution on b_{sat} and a given training runtime model. The second difference is that

[Provost et al \(1999\)](#) check whether the saturation point has been reached using a method called *linear regression with local sampling* (LRLS), which samples not only *at* but also closely *around* anchors to estimate the slope at an anchor and stop if the slope is close to zero.

The practical benefit of the LRLS scheme is not entirely clear for sample-wise curves. First, since the slope is also based on empirical values, from a theoretical viewpoint, it is not clear that the criterion is necessarily better than the naive approach suggested by [John and Langley \(1996\)](#). Second, the approach is more expensive than the naive approach because more observations need to be sampled; this can be a substantial factor, especially for large anchors. LRLS becomes relevant when applied to learners that have a computational complexity for training that scales worse than linear in the number of training points (e.g., Gaussian processes or decision trees). For learners with training complexity linear in the number of observations, it will often be more expensive than evaluating a learner's performance on the complete dataset. This can be seen with a simple calculation, in which we assume roughly linear training time complexity: Suppose a costly anchor at 40% of the overall data size. If we draw only two additional samples around this anchor, then the runtime is around 120% of the runtime we would have had if we had trained on the complete dataset once. While [Provost et al \(1999\)](#) argue that LRLS only adds a constant factor to the runtime, [Sarkar et al \(2015\)](#) reasonably argue that this factor is often prohibitive in practice.

The approaches in this section explicitly assume that more data than b_{sat} is available. This implies that they can be used primarily for early stopping scenarios rather than data acquisition scenarios. Still, if a learner does not attain saturation performance on the complete dataset, the approaches can detect this at the cost of the additional evaluations at the non-final anchors.

Concerning the stability of estimates, [Beleites et al \(2013\)](#) point out the necessity to have an estimate for the *confidence interval* not only for the performance at the anchors in the training schedule but also on the *test data*. They argue that confidence intervals are essential when deciding whether reasonable generalisation statements can be made for a classifier. This changes the notion of the stopping point to, perhaps, a *confident* stopping point. The optimal stopping point may be reached early, but the validation fold sizes may still be too small to assure stable assertions. Typically, the confidence intervals are large on small anchors and then contract for increasing anchor sizes. Based on credible intervals, the authors propose choosing the anchor point that achieves a sufficiently narrow interval on the test data.

[Ng and Dash \(2006\)](#) address the impact of class imbalance on the performance of a learner. That approach hypothesises that, without further knowledge, the class distribution in which all classes have the same number of observations is optimal. The authors modified the aforementioned progressive sampling scheme by creating train sets at each anchor such that all classes have the same distribution. For anchors of sizes that would require more instances of a class than are available in the existing data, random instances of that class

are replicated until the class balance is established again. Note that while the work is based on the findings by Weiss and Provost (2003) (cf. Sec. 2.5.2), they apply a different strategy. Instead of optimising over the class distribution for a given anchor size, they try to find the stopping point under the premise that the training set will always be balanced.

Regarding the stopping point of iteration-wise curves, a common technique is to separate some data that is not used for training but to compute an iteration-wise curve online to detect convergence (Bishop, 1995).

Projective Approaches

A different idea to obtain the saturation point b_{sat} was proposed by Leite and Brazdil (2003, 2004) through the notion of metalearning (Brazdil et al, 2022). Similar to Provost et al (1999), a geometric schedule is used. The assumption is that we already know the performances of the current learner at all anchors in the schedule on *other* datasets. The idea is to compute, on the target dataset, the performances only for the very first anchors and then to *predict* the saturation point by aggregating the (known) saturation point on the *k most similar* learning curves of the other datasets. The distance measure here is the sum of differences between the curves at the initial anchors; the concrete anchors used in their paper are (91, 128, 181, 256, …), corresponding to the powers of $\sqrt{2}$. The authors consider different aggregation measures such as mean and minimum (Leite and Brazdil, 2003) and the median (Leite and Brazdil, 2004).

The authors discuss the potential issue that, among the k nearest neighbour curves, some or even all of the curves can be substantially different from the partial learning curve on the target dataset. Using the k nearest learning curves to predict the stopping point would not work in such cases. Follow-up work (Leite and Brazdil, 2007) proposes a remedy to this problem, in which the curves are not used directly but are adjusted via a concept called *curve adaptation* (discussed in Sec. 5.1.6).

5.1.5 Finding the Utility-Based Stopping Point

The problem of identifying the utility-based stopping point was, to our knowledge, first addressed by Meek et al (2002) and was also independently investigated by Weiss and Tian (2006). In these papers, a retrospective approach is applied. The idea is similar to the aforementioned concept of progressive sampling (Provost et al, 1999), except that the analysis is done for utility rather than learning curves. In contrast to the learning curve, the utility curve does not plateau but starts to deteriorate after its peak (see Fig. 4).

The main difficulty with the concept of utility in the context of learning curves is to find a unifying scale for (i) the costs of data acquisition and training time and (ii) the model performance. Meek et al (2002) avoid this problem by adopting the notion of implicit utility through the comparison with a baseline. They stop the algorithm when the ratio between the benefit improvement and the augmented runtime drops below a pre-defined threshold.

In contrast, [Weiss and Tian \(2006\)](#) compute an explicit utility, and the algorithm stops as soon as the *observed* utility decreases for the first time, which is taken as the indicator that the utility-based stopping point has been passed. The authors adopt the concept of the net utility of a potential classifier, which is the difference (in utility) between predictive performance (under a hypothetical number of training instances) and the cost to acquire the (additional) instances. Notably, the assumption is that the user has no control over the instances for which labels will be acquired next, which contrasts the approach from *active learning*. To merge different types of inconveniences (i.e., acquisition costs and prediction errors) into a single utility measure, the user has to *define* costs per unit, e.g., costs for acquiring a single usable training instance and costs for making a wrong prediction. Furthermore, the authors consider the problem of deciding *online* whether or not to acquire more data and, in the affirmative case, how *many* instances should be considered in the acquisition batch before reconsidering. The latter effectively corresponds to deciding upon a progressive sampling scheme ([Provost et al, 1999](#)). However, the paper does not analyse the effects of fixed costs per batch (such as the computational costs of training a model), which implies that one could set batch sizes to 1 without consequence.

A consecutive version of that paper also adds the CPU cost for model induction to the costs of a point on the learning curve ([Weiss and Tian, 2008](#)). The original paper only considered acquisition costs and the prediction error. This model also seems suitable for iteration-wise curves, where there would be no data acquisition costs.

In all of the above approaches, the usage of the empirically gathered learning curves for decision making is minimal. Moreover, the approaches ignore all except the *last two* points on the learning curve. In this sense, and in terms of the stopping point approaches, the above works are retrospective in nature. No model of the learning curve is built, and no projections of errors on bigger training sizes are made, which, for example, could make sense to predict that this utility peak event will occur in the future or even only in the next iteration. [Last \(2007\)](#) proposes such an approach, which we discuss in Sec. 5.2.2.

5.1.6 Performance Prediction at Fixed Point

The problem of predicting the learning curve value is naturally a regression problem, where the goal is to predict $\mu_{a,b}$ for a fixed budget b . Essentially, given a fixed learner a and the budget expressed in either observations n or iterations t , it is about predicting $\mathcal{C}(a, n)$ for sample-wise curves, and predicting $\mathcal{C}(a, n, t)$ for sample-wise curves. The attributes are the performance values at different (cheap) anchors and potentially additional contextualising attributes. Using these attributes, one explicitly or implicitly generalises over *datasets* or *learning algorithms* (or both). Models that generalise over datasets are typically called meta-models and rather aim at model ranking ([Brazdil et al, 2022; Ruhkopf et al, 2023](#)), which we discuss in Sec. 5.1.2. This is because

it is difficult to generalise exact performance across datasets. When generalising over learners, one typically trains a single model for the target dataset, trained from learning curves on the same dataset belonging to other learning algorithms. Models that generalise over learners are typically called surrogate models (Eggensperger et al. 2018).

We organise this section by how explicit the generalisation is made over one of the two concepts. If no additional attributes are available, one implicitly assumes that all entries in the database are to a degree suitable to predict values in a new situation. There is no explicit context in this case, and we are generalising implicitly over datasets or learners. Since no explicit contextualisation exists, those approaches can be used for both purposes, regardless of their original purpose. In contrast, additional contextualising attributes can describe the dataset, i.e., we have meta-features of the datasets available, or they can describe the learning algorithms to which the learning curve values belong. In principle, one could utilise both types of additional attributes, but we are not aware of any approaches that adopt both.

Generalization Without Explicit Context

Generalisation from learning curves without explicit context means to predict the performance of a given learner on some dataset based on previously acquired learning curves that are not equipped with additional information, i.e., features describing the dataset or the algorithm used to produce them. To justify the prediction model, the existing empirical learning curves either stem from the same algorithm on other datasets, or other algorithms on the same dataset. Either of these implicitly qualifies them to be relevant for the new task. In other words, one simply uses a set of unannotated existing learning curves as aids to predict the behaviour of a new, only partially known, learning curve.

[Chandrashekaran and Lane \(2017\)](#) developed an approach that explicitly used regression to predict the target performance without context. Probably without noticing, the approach mainly re-invents the approaches previously developed by [Leite and Brazdil \(2005, 2007, 2010\)](#); [van Rijn et al \(2015\)](#), since it computes the most similar other learning curves in the portfolio and obtains a prediction based on the average over those curves. The three differences are that [Chandrashekaran and Lane \(2017\)](#) (i) consider uncertainty in the prediction based on the variance in the neighbourhood, (ii) adopt an affine transformation (instead of a linear transformation) of the existing learning curves and apply this *before* selecting the most similar ones, and (iii) that they do not use a fixed schedule but, due to the focus on iteration-wise curves, simply a continuous schedule that is stopped as soon as there is enough evidence that the target performance will not be better than a current threshold. The Euclidean norm between the vectors describing the performances at the anchors is used as the distance function between two curves.

The reason why the approach is discussed here and not in Sec. 5.1.2 together with the others is subtle and worth being discussed. In both lines of research,

target performance values from related curves are averaged to estimate the performance of the current learner. However, [Chandrashekaran and Lane \(2017\)](#) explicitly treat this as a performance prediction, which is then used for early discarding by comparing against a threshold. In contrast, the works in the line of [Leite and Brazdil \(2005\)](#) do not treat this value as an actual prediction but simply as some score used to order a pair or a set of learners.

While the approach generalises across algorithm configurations, it ignores the configurations and generalises from learning curves without explicit context. In other words, there is no reason why the approach could not be used also for generalisation across datasets.

[Cardona-Escobar et al \(2017\)](#) presented an approach that predicts values at *all* anchors. The authors adopt a series of support vector regression models, one for each anchor not evaluated so far. It is not entirely clear with which data the models are trained, but we presume that it follows the same logic as [Chandrashekaran and Lane \(2017\)](#) and uses the fully known learning curves of previously evaluated neural network configurations to do so. Interestingly, similar to chaining ([Gkioxari et al, 2016](#)) in classification, they use as inputs for the j -th future anchor not only the known partial learning curve values but also use the *predictions* for the anchor points predicted before j .

Generalization With an Explicit Algorithm Context

Generalisation across algorithms only considers the target dataset and assumes that a number of (complete) empirical learning curves on that dataset are already available for different algorithms. The *explicit* generalisation requires that the previous learning curves are *explicitly* associated with features describing the algorithm to which they belong.

[Baker et al \(2018\)](#) propose a method that uses features describing *both* the learning curve (including up to second-order differences) *and* the algorithm. The approach predicts the performance of neural networks based on features that describe the architecture (number of layers and weights) as well as the hyperparameters of the learning algorithm (such as learning rate, learning rate decay, etc.). They adopt linear and kernel-based support vector regression machines, random forests, and simple linear regression based on ordinary least squares. Even though the authors suggest using kernel-based support vector regression machines, they find that simple linear regression also often compares highly competitive for this prediction task.

Following this idea, [Long et al \(2020\)](#) additionally add textual descriptions of the architecture to predict the learning curves of neural networks. Indeed, the architecture description by [Baker et al \(2018\)](#) is rather simplistic and only immediately well suited if all layers are of the same type, e.g., dense layers, and have the same number of neurons. [Long et al \(2020\)](#) report substantial performance improvements for convolutional network architectures compared to the approach taken by [Baker et al \(2018\)](#).

5.1.7 Performance Bounding

Performance bounding tries to give explicit lower or upper bounds on the performance value at some specific or arbitrary budget. Answering this question is essential to make high-confidence decisions on early discarding as discussed in Sec. 4.1.3; formally, we denote it as $\underline{\mathcal{C}}(a, \cdot)$. Typically, one is interested in $\underline{\mathcal{C}}(a, b_{ref})$, b_{ref} being the size of the dataset intended for training or the limit performance of an iteration-wise curve, but the models we discuss here are more general.

Performance bounding is intuitively a simpler problem than performance prediction (Sec. 5.1.6), but there are usually also higher expectations with respect to the accuracy of the assertion. When a performance bound is expressed, one would expect that the true value is at least as high as specified with high probability.

One approach that addresses this problem is Data Allocation using Upper Bounds (DAUB) (Sabharwal et al., 2016). Given a set of configurations, it first runs all configurations on two anchors of the dataset, effectively building the initial segment of the learning curve. Based on this initial segment per configuration, it determines an optimistic performance bound for each learning curve that likely will not exceed (i.e., an upper bound for measures that need to be maximised, such as accuracy, and a lower bound for measures that need to be minimised, such as error rate). This performance bound is determined by calculating the linear regression slope of the last two segments. The performance on the last anchor is extrapolated to the full size of the dataset according to this slope. Therefore, there is an optimistic upper bound on the performance that a configuration can obtain. After that, it goes into the following loop: It runs the most promising configuration on a larger sample size and updates the performance bound. It reevaluates which configuration has the most potential at that budget and assigns more budget to the most promising configuration until one configuration has been run on the entire dataset. Therefore, this is an example of horizontal model selection.

Alternatively, learning curve cross-validation uses a similar approach but addresses this in a more flexible, vertical setting (Mohr and van Rijn, 2021, 2023). The method explicitly assumes that sample-wise curves have a convex behaviour. The convexity of the curve allows for deriving a best-case extrapolation from a partial empirical learning curve. The convexity assumption is used to linearly extrapolate the empirical learning curve and prune learners when they can no longer improve on the best-known solution.

5.2 Approaches With Learning Curve Models

This section covers learning curve approaches that utilise an explicit model to model the entire learning curve, as described in Sec. 3. Fig. 14 shows an overview of all the approaches we discuss and how they cite each other. In the base form, there is a model for a specific learner a that is able to predict the performance $\mathcal{C}(a, \cdot)$ or $\mathcal{C}(a, n, \cdot)$ at any sample size or iteration respectively

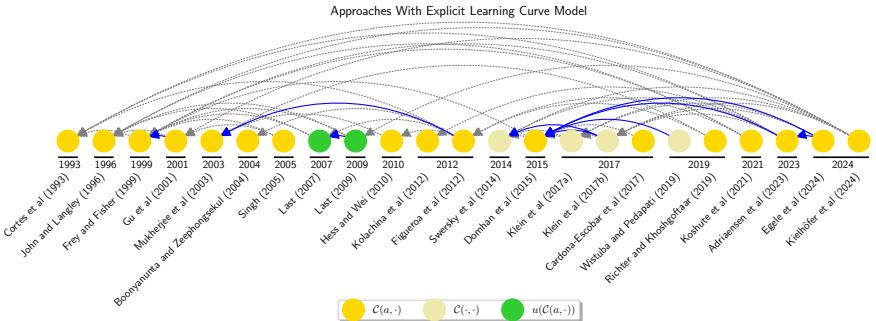


Fig. 14: Citations across approaches with a learning curve model. The colours indicate the question addressed about learning curves. Normal arrows indicate that the paper cited the other paper, and coloured arrows indicate an experimental comparison with previous approaches.

(yellow), discussed in Sec. 5.2.1. On top of such curves, a utility curve \mathcal{U} can be defined (green), discussed in Sec. 5.2.2. Finally, the curve models can even generalise across the learners, leading to generalised curve models $\mathcal{C}(\cdot, \cdot)$ (beige). One additional benefit of these types of models is that they can also perform model acquisition, i.e., assess the performance of a learner for which we have no performance evaluations yet. These models are discussed in Sec. 5.2.3.

In this section, we describe the learning curve model according to the notion of *budget*, using the variable b to avoid having to distinguish between sample sizes n or iterations t . In the spirit of Sec. 3, we use the notation $f(a, b)$ to refer to the original random variable that generates observations of the performance of learner a at budget b (independently of whether f here is a sample-wise curve with b being the sample size or whether f is an iteration-wise curve with some implicit sample size n and b being the iteration). Accordingly, $\mu_{a,b} = \mathbb{E}[f(a, b)]$ is the mean value of the curve of learner a at budget b .

5.2.1 Performance Prediction at Any Point

We will discuss approaches that build a full learning curve model $\hat{f}_a(b)$ for a specific learner a . That is, they model the curve mean $\mu_{a,b}$ for a fixed learner a at *any* budget b (effectively modelling $\mu_{a,\cdot}$). We divide the approaches into three groups, corresponding to the three model types for \hat{f} discussed in Sec. 3, which are also reflected in the three layers of Fig. 10. Accordingly, we first discuss approaches that provide *point estimates* of the curve, i.e., $\hat{f}_a : \mathbb{N} \rightarrow \mathbb{R}$, then approaches that explicitly treat those estimates with uncertainty and introduce a notion of bounds on them, i.e., $\hat{f}_a : \mathbb{N} \rightarrow \mathbb{R}^2$, and finally approaches that create entire probabilistic belief models over curves, i.e., $\hat{f}_a : \mathbb{N} \rightarrow \{p \mid p \text{ is a distribution of a real-valued random variable}\}$.

Point Estimates

To our knowledge, the first approach that used observed data to fit a learning curve model was presented by Cortes et al (1993). In that paper, the three-parametric inverse power law shown in Eq. (6) was used to build an iteration-wise curve. The usage of the power law model is justified with the findings in the statistical mechanic framework (Seung et al, 1992) and used to predict the predictive performance on the complete dataset (for two neural network architectures on the NIST dataset). The authors find that the predictive performance on 60k instances can be almost perfectly predicted using the inverse power law. Unfortunately, it is not entirely clear on which anchors the estimates are built. Notably, this is the only paper we are aware of that fits *both* a train- and test-error curve. The paper reports some information about noise through boxplots on the curve but does not explicitly incorporate them into the model.

John and Langley (1996) proposed a similar approach. Similar to the work of Cortes et al (1993), the model is used to estimate the performance of a learner on the complete dataset with the goal of stopping the training procedure early. One difference is that the model is adopted for a sample-wise curve instead of an iteration-wise curve. Concerning convergence detection, John and Langley (1996) employ the Probably Close Enough criterion, which detects convergence if the probability that the accuracy of a model trained on the complete dataset will be at most some ε worse than the current model's performance is less than some δ . However, the paper itself then does not adopt a notion of probabilities but stops if the performance on the complete dataset predicted by an inverse power law model is not by at least ε better than the currently observed performance; they call the approach Extrapolation of Learning Curves. In other words, the uncertainty is not quantified.

The inverse power law has been used in many applications. For example, several approaches have used the inverse power law to model the performance of neural networks in different domains (Alwosheel et al, 2018; Cho et al, 2015). It is noteworthy that recent works show evidence against the usage of any commonly used models for neural networks, at least in the initial parts of the curve, due to the (sample-wise) double descent (Nakkiran et al, 2020). At least for certain combinations of architectures, datasets, and training procedures, there is empirical evidence that the learning curve exhibits non-monotonic behaviour, which contradicts all existing learning curve models like the inverse power law model.

Range Estimates

Mukherjee et al (2003) built inverse power law models in the domain of DNA data. The main contribution of that paper is to analyse the appropriateness of the inverse power model on eight medical datasets. To this end, they construct uncertainty bounds around the mean learning curve consisting of the q25 and q75 curves fitted from those statistics, respectively. This way, a learning curve model including information about dispersion is obtained. Experiments are

conducted for a support vector machine on eight medical datasets in which the leave-one-out validation result (estimate of the learning curve on $|d| - 1$ data points for training) is compared to the boundaries suggested by the model. Having an explicit model for the q25 and q75 curves, one can obtain for an arbitrary anchor b not only a point estimate of $\mu_{a,b}$ but an estimate of the inter-quartile range of $f(a, b)$ itself, which is arguably more informative given that, assuming Gaussian noise, the interval should also contain $\mu_{a,b}$.

[Figueroa et al \(2012\)](#) modify the aforementioned approach in two ways. First, different anchors are associated with different *weights*, usually to assign higher weights to larger anchors since they are more informative. Second, implicitly assuming a standard Gaussian distribution of the observations as in Eq. (5), they compute a 95% confidence interval to describe the uncertainty rather than the interquartile range. Based on this information, for a query point b , they predict a confidence interval instead of a point estimate. [Figueroa et al \(2012\)](#) also applied this approach to medical data, just as [Mao et al \(2016\)](#) for EEG data. More recently, it was also successfully used for sensor communication ([Oyedare and Park, 2019](#)). This work aims to predict a reasonable sample size, which is perhaps more reasonably addressed by the utility-based approaches discussed in Sec. 5.1.5.

Recently, [Koshute et al \(2021\)](#) have used the inverse power law to predict the minimum anchor point on which a learner must be trained to reach near- p_{sat} performance with a *given desired* confidence. This approach can be seen as a combination of the above two approaches. Similar to [Figueroa et al \(2012\)](#), they compute the confidence interval at all anchors. However, instead of using these to estimate confidence intervals at arbitrary points, they fit a single curve on the lower bounds of the confidence intervals at the known anchors. The resulting model is not used to make predictions on arbitrary anchors but to compute the cheapest anchor that will obtain with a pre-defined probability (size of confidence-interval) a performance that is ε -close to p_{sat} , where ε is a hyperparameter controlled by the user.

The idea of computing confidence intervals is also adopted by learning curve cross-validation ([Mohr and van Rijn, 2021, 2023](#)). A Morgan-Mercer-Flodin model is created to decide whether or not to skip intermediate anchors and evaluate the learner on the full dataset size. However, the confidence intervals are used differently than in the above cases and are not used for the inverse power law model itself. In contrast, the confidence bounds are used to compute the range of possible *slopes* of the learning curve between two anchors.

Distribution Estimates

The first approach to predict distribution estimates for any anchor point was presented by [Domhan et al \(2015\)](#). The approach assumes learning curves to be instances of a parametric model that is a *linear combination* of known model classes, such as the inverse power law, and others ([Gu et al, 2001](#)). The main difference to the above approaches is that, instead of estimating the parameters through a maximum likelihood approach, they estimate, for each

parameter, the whole posterior *distribution* adopting Monte-Carlo Markov Chains (MCMC). The approach is successfully used to early discard neural network architectures by predicting the saturation performance p_{sat} of an iteration-wise curve, thereby discarding learners as soon as the probability that it is competitive drops below a pre-defined threshold. The Bayesian model proposed by Domhan et al (2015) also explicitly estimates the noise $\sigma_{a,b}^2$, which is assumed to be homoscedastic, i.e., identical for all budgets b . In this sense, the approach quantifies *both* epistemic and aleatoric uncertainty (cf. Sec. 3).

A recent alternative for estimating distributions has been proposed by Adriaensen et al (2023) through the notion of prior fitted neural networks (PFN) (Hollmann et al, 2023). Based on the approach by Domhan et al (2015), Adriaensen et al (2023) consider a set of basis functions, over which they define a prior. They then sample a large number of curves from this prior and train a variation of a transformer neural network with it, which is able to predict *distributions* for a target anchor based on a partial learning curve. Since no sampling from the posterior is required anymore, compared to the MCMC approach by Domhan et al (2015), predictions can be obtained much faster, and the authors claim that the prediction performance is comparable or even better, which could however not be confirmed in subsequent experiments discussed below.

The latest approach we are aware of is the robust estimation RoBER presented by Egele et al (2024). The approach follows the idea of MCMC introduced by Domhan et al (2015) but applies a different sampling algorithm to obtain more stable estimates. The results suggest that classical learning extrapolation significantly outperforms PFN-based extrapolations at the current state of research.

5.2.2 Utility Prediction at Any Point

Utility prediction combines learning curve models as discussed in Sec. 5.2.1 with utility models as discussed in Sec. 5.1.5. The learning curve model f is used as a basis to estimate the performance at any point, and the utility at budget b is then computed as a function of the modelled performance $\hat{f}_a(b)$ at and the associated costs for budget b .

The first approach in this direction was presented by Last (2007). This work is very similar to the works of Weiss and Tian (2006) but makes utility *forecasts* rather than looking back. Therefore, it is projective instead of retrospective. The error rate that serves as input to the model is obtained from a parametric model (i.e., a power-law) trained on the empirical values obtained at earlier budgets. This framework enables one to *analytically* compute the optimal dataset size. The main advantage of the approach over the one of Weiss and Tian (2006) is that one does not need to go through several acquisition iterations, which is a benefit if those are associated with fixed costs. Therefore, the learning curve has become a resource for decision making. While this work assumes the empirical learning curve to be available, Last

(2009) embeds his idea into an algorithm that follows a progressive sampling scheme in a follow-up work.

One use case in which the above techniques have been adopted has been reported in the context of automated software configuration (Sarkar et al, 2015). The context of that paper is that every instance is a parametrisation of a software library, and obtaining its label requires the costly execution of a benchmark on such a configuration. The goal is to understand how many observations must be acquired to learn a reliable prediction model. To this end, the authors adopt the projective sampling approach of Last (2007, 2009).

The latter work raises an important issue by stating that knowing the saturation point is (often) not enough. Instead, we often also *need* to know the performance at the saturation point. Sarkar et al (2015) argue that if the user is unaware of the expected performance at that point, substantial resources might be required to obtain the observations to reach the saturation point. However, if the actual performance at that point is known to be mediocre, the user could anticipate this and not invest the required resources. To this end, they also incorporate the utility model proposed by Weiss and Tian (2008).

5.2.3 Performance at Any Point for Any Learner

The approaches discussed in this section are the most general ones developed to date regarding learning curves in that they create a model for the complete function \mathcal{C} , i.e., generalising both over both budgets and learners. Such a model is so versatile that it can be used in *all* types of decision-making situations, e.g., data acquisition, early stopping, and early discarding. Additionally, these can be used for model acquisition (selecting a yet unseen promising model).

Freeze-thaw Bayesian optimisation models the behaviour of learning curves through Gaussian processes (Swersky et al, 2014). An important contribution of that work is a non-stationary kernel for Gaussian processes that supports exponentially decaying learning curve models; it can easily be checked that standard kernels like a linear or Gaussian kernel do not lead to meaningful learning curve models. Assuming that the kernel reflects the model class appropriately, one additional benefit of using a Gaussian process is that one automatically obtains estimates for the noise $\sigma_{a,b}^2$ at an arbitrary anchor b . Using their kernel and the current set of observations, Swersky et al (2014) estimate the asymptotic mean performance p_{sat} . Since the learning curves are combined with Bayesian optimisation, the uncertainty for a specific future anchor is one of the required inputs for computing their acquisition function. In a rather thin evaluation, the approach was successfully applied to Online Latent Dirichlet Allocation, Logistic Regression, and Probabilistic Matrix Factorization, considering one dataset per learner. While the paper focused on iteration-wise curves, the modelling technique can also be used for sample-wise curves.

Klein et al (2017a) presented a similar approach dubbed FABOLAS. Similar to Freeze-thaw Bayesian optimisation, a Gaussian process is used to model

the learning curve of the learners across different hyperparameter configurations. There are three main differences between the two approaches as far as learning curves are concerned. First, FABOLAS considers sample-wise curves, while freeze-thaw Bayesian optimisation considers iteration-wise curves. Second, and related to this, FABOLAS uses a kernel different from freeze-thaw Bayesian optimisation to model the behaviour of the learning curve using the Gaussian process, which is defined by the relative dataset size in $[0, 1]$ instead of absolute sizes. Third, FABOLAS tries to explicitly learn the complete learning curve, while freeze-thaw Bayesian optimisation focuses only on the saturation performance. Similar to freeze-thaw Bayesian optimisation, the uncertainty about the performance estimates of the learning curve is not explicitly used. Still, the fact that a Gaussian process is fitted from the data allows one to make assertions about the certainty of the learning curve value at any point.

Parallel to their work on FABOLAS, Klein et al (2017b) proposed an approach to estimate both the mean and the *noise* (aleatoric uncertainty) of a learning curve through the notion of Bayesian neural networks. The neural network predicts the parameters of a set of basis functions. These basis functions incorporate prior knowledge into the network, which is necessary to extrapolate away from the data. The main difference between this approach and the aforementioned approaches is that this approach models the behaviour of the learning curve through a neural network. This network has $d + 1$ input units (d for the algorithm description and one for the anchor), one output unit for the estimated performance and, optionally, one output unit for the estimate of the variance of the performance (which relates to aleatoric uncertainty). To our knowledge, this approach and the learning curve extrapolation proposed by Domhan et al (2015) are the only approaches in this area that explicitly model the variance of the performance of the learning curve (i.e., which can also be seen as noise). An important difference between the two is that Klein et al (2017b) assume *heteroscedastic noise*, i.e., noise that changes with both different hyperparameters and anchors, while Domhan et al (2015) assume homoscedastic noise across anchor sizes (not across configurations, because the model does not generalise over different configurations). While the approach presented in the paper does not explicitly consider the uncertainty about the parameter estimates, the parameters are essentially sampled from a posterior distribution. Therefore, the uncertainty is at least implicitly available. However, it should be noted that the number of parameters describing the model here, namely the network weights, is potentially *much* larger than in the approach taken by Domhan et al (2015).

Wistuba and Pedapati (2019) propose to use biased matrix factorisation to model $\mathcal{C}(\cdot, \cdot)$. The approach is settled in the context of neural architecture search. Knowledge from previous datasets and different architectures is used to estimate the performance of new architectures on the target dataset, and this estimate is used to drive a Bayesian optimisation approach.

Table 1: Overview of the discussed learning curve approaches, ordered along the most general question they address, the learning curve type and the data resources used (4 columns). In the header, ‘LC’ stands for learning curve, ‘DS’ stands for dataset, and ‘AL’ stands for algorithm. All of them use partial empirical curves on the target dataset of the current learner(s). Estimate type: p,r,d are *point*, *range* and *distribution* estimates, respectively.

Question	Type	LC	other DS	DS Meta-Feat.	LC other AL	AL Meta-Feat.	Utility	Estimate Type	Contributions
$p_{ref} > \tau$	obs.	X	X	X	X	X	X	p	Van den Bosch (2004); Jamieson and Talwalkar (2016); Petрак (2000); Zeng and Luo (2017)
$p_{ref} > \tau$	both	X	X	X	X	X	X	p	Li et al (2017)
$\pi_{a \in A}$	obs.	✓	X	✓	X	X		p	Leite and Brazdil (2005, 2007); van Rijn et al (2015)
$\pi_{a \in A}$	obs.	✓	✓	X	X	X		p	Leite and Brazdil (2008, 2010)
$\pi_{a \in A}$	both	✓	✓	X	X	X		p	Ruhkorf et al (2023)
b_{sat}	iter.	X	X	X	X	X		p	Bishop (1995)
b_{sat}	obs.	X	X	X	X	X		p	John and Langley (1996); Ng and Dash (2006); Provost et al (1999)
b_{sat}	obs.	✓	X	X	X	X		p	Leite and Brazdil (2003, 2004)
p_{lim}	obs.	X	X	X	X	X		p	Cortes et al (1994)
b_{sat}^u	obs.	X	X	X	X	X		p	Meek et al (2002)
b_{sat}^u	obs.	X	X	X	X	X	✓	p	Weiss and Tian (2006, 2008)
$\mathcal{C}(a, d_{tr})$	obs.	✓	X	X	X	X		r	Chandrashekaran and Lane (2017)
$\mathcal{C}(a, d_{tr})$	iter.	X	X	✓	✓	X		p	Baker et al (2018)
$\mathcal{C}(a, \cdot)$	obs.	X	X	X	X	X		p	Sabharwal et al (2016)
$\bar{\mathcal{C}}(a, \cdot)$	obs.	X	X	X	X	X		r	Mohr and van Rijn (2023)
$\bar{\mathcal{C}}(a, \cdot)$	iter.	X	X	X	X	X		p	Cortes et al (1993)
$\mathcal{C}(a, \cdot)$	obs.	X	X	X	X	X		p	Boonyanunta and Zeephongsekul (2004); Frey and Fisher (1999); Gu et al (2001); Hess and Wei (2010); John and Langley (1996); Kolachina et al (2012); Richter and Khoshgoftaar (2019); Singh (2005)
$\mathcal{C}(a, \cdot)$	obs.	X	X	X	X	X		r	Figueroa et al (2012); Koshute et al (2021); Mukherjee et al (2003)
$\mathcal{C}(a, \cdot)$	iter.	X	X	X	X	X		d	Domhan et al (2015)
$\mathcal{C}(a, \cdot)$	obs.	✓	X	X	X	X		p	Cardona-Escobar et al (2017)
$\mathcal{C}(a, \cdot)$	both	✓	X	X	X	X		d	Adriaensen et al (2023)
$\mathcal{C}(a, \cdot)$	both	X	X	X	X	X		d	Egele et al (2024)
$\mathcal{C}(a, \cdot)$	obs.	✓	✓	X	X	X		p	Kielhöfer et al (2024)
$\mathcal{U}(a, \cdot)$	obs.	X	X	X	X	X	✓	p	Last (2007, 2009)
$\mathcal{C}(\cdot, \cdot)$	both	X	X	✓	✓	X		d	Klein et al (2017b); Swersky et al (2014)
$\mathcal{C}(\cdot, \cdot)$	obs.	X	X	✓	✓	X		d	Klein et al (2017a)
$\mathcal{C}(\cdot, \cdot)$	both	✓	X	✓	✓	X		d	Wistuba and Pedapati (2019)

6 Summary and Open Research Directions

Learning curves have been a vital resource for decision making in machine learning for several decades, and they have gained significant attention over the last years. Learning curves have proven to be a suitable solution for different

types of decision-making situations, i.e., data acquisition, early stopping, and early discarding for model selection.

We have provided a formal definition of various types of learning curves (Sec. 2). There are two predominant types of learning curves in the machine learning literature, i.e.: the sample-wise curve (i.e., the type of learning curve that one obtains when giving a learner more training instances) and the iteration-wise curve (i.e., the type of learning curve that one obtains when allowing an algorithm to process the data multiple times, see for example the number of epochs of a neural network). While both types of learning curves seem similar, they have distinct semantic meanings and characteristics. Both types of learning curves can be extended to a utility curve, which considers the cost of computational resources or data acquisition. Additionally, we have contrasted these against other types of (learning) curves, such as feature curves, capacity curves, and curves obtained by data-centric models, such as active learning or curriculum learning.

We have described the basic concepts of modelling a learning curve (Sec. 3). There are various parametric models that incorporate domain knowledge about what we already know about the shape of learning curves (e.g., the three-parameter inverse power law-model). Even when the performance of a learner is observed at very few anchors, the learning curve can already be extrapolated to make predictions about larger anchors. Additionally, one can decide to also model a degree of uncertainty, either as a range estimate or as a distribution. We distinguish between two types of uncertainty, i.e., epistemic uncertainty and aleatoric uncertainty, and relate these concepts to the literature on modelling learning curves. Typically, when uncertainty is being modelled, the epistemic uncertainty is being modelled, but in some cases, the aleatoric uncertainty is being modelled (see, e.g., Klein et al, 2017b).

We have provided a unified framework for methods that utilise learning curves for decision making in machine learning (Sec. 4). This framework categorises these methods along three axes: the decision situation that they address, the questions that can be addressed with learning curves, and the data resources that can be used to model the learning curves. Notably, Fig. 10 shows an overview of all questions that can be addressed by learning curves. There are various ways to address decision situations with learning curves; for example, questions about the saturation point of a given learner or whether a learner will perform better than another learner at a given amount of data. These questions can be further generalised, eventually ranging in complexity from binary questions to questions that address how any learner behaves at any budget.

We have done an extensive literature survey, categorising all learning curve methods that we are aware of into this framework (Sec. 5). Table 1 shows an overview of the methods we have discussed in this survey, contextualising them according to these criteria. This table can be seen as an extension of Fig. 12. Based on this literature survey, we describe several directions for future work.

More experimental databases for learning curve research to support the full complexity of learning curve methods. Doing relevant research on learning curves requires extensive computing power. Exploring a sample-wise curve inherently requires many models to restart the learning process at different anchors, whereas exploring an iteration-wise curve is often done on neural networks that come with their distinct layers of complexity (see, e.g., [White et al, 2023](#)). A common way to address this is by experimental databases or surrogate benchmarks that store certain experimental results. This allows for fast experimentation and, therefore, faster development cycles. While several of these experimental databases for learning curves exist (as outlined in Sec. 2.2), these currently do not capture the full scale of learning curves resources or questions that can be answered using learning curves.

A quantitative benchmark on learning curve extrapolation methods. Many models have been proposed to extrapolate learning curves and make predictions about the performance of a learning at a higher budget (see, e.g., [Gu et al, 2001](#)). However, these models have only been subject to limited comparison. While [Gu et al \(2001\)](#) compared various parametric models against each other, and [Kielhöfer et al \(2024\)](#) compared a representative parametric model against a representative metalearning model across many different settings, more research is needed. Fig. 13 and Fig. 14 already show that, while many papers are aware of other methods and cite those methods (grey arrows), only very few actively compare against each other (blue arrows). Moreover, many of these learning curve models are used as a small component in a larger system, e.g., an AutoML system. In such a case, the predictive performance of the learning curve model might not even be measured, as eventually, one often measures the quality of the complete system; in the case of an AutoML system, the performance of the final selected model. Due to this modular nature, improvements on the learning curve extrapolation method would then be orthogonal to improvements on the AutoML system.

Tighter integration of learning curve extrapolation methods with AutoML systems. We already noted a clear opportunity for AutoML systems in Sec. 4.1.3. In situations where multiple learners are being compared against each other, the training set needs to be further split into an actual training set and a validation set to select the best learner to be tested on the test set (which can only be seen once). The existence of this validation set already shows an opportunity for learning curve methods; while a learner is being selected based on its performance after being trained on the split-off training set, what is relevant is its performance after being trained on the original training set (i.e., the split-off training set plus validation set). It is not necessarily the case that the same learner performs best on both. Learning curve extrapolation models can predict which learner will eventually perform best on an anchor of the size of the original training set.

More learning curve methods that operate on low-level questions. A prominent question that arises from the literature survey is whether simple questions can be treated more simplistically. Fig. 10 shows four binary

questions. While approaches aim to answer these questions, most do so by implicitly answering a more difficult question (see Table 1 and Fig. 12). Some of these questions are really at the core of the discussed approaches. For example, will the learning curve intersect with the learning curve of the currently best model? Most approaches create a learning curve model for this, implicitly solving a much more difficult problem. While those models, if appropriate, have the potential to provide additional interesting insights, the question arises whether simpler approaches could reliably solve those problems while needing much less online data. Approaches that remain faithful to this question level (such as successive halving and hyperband) have proven effective and received considerable attention. Additionally, we see a clear opportunity for incorporating uncertainty into methods that address the binary question, effectively providing the chance that a particular learner will be better than another learning at a given budget.

A learning curve method that makes use of all types of data resources. Learning curve methods can make use of various resources to model the learning curve (see Fig 11). For example, the inverse power law model uses the current learner's learning curve on the same dataset. In contrast, metalearning models often also make use of learning curves of either the current learner or other learners on other datasets (see, e.g., Leite and Brazdil, 2005). A reasonable assumption is that the methods that utilise more types of data resources would be more accurate. Table 1 reveals that no learning curve model utilises all types of data resources. Indeed, combining anchors across learners and datasets might be a complex task, but when this is done successfully, it will enormously increase our understanding of learning curves.

Acknowledgements.

We thank Pavel Brazdil, Isabelle Guyon, Aaron Klein, and Marco Loog for their insightful discussions and feedback on this survey.

A Notation

Table 2 contains an overview of the notation used throughout this paper.

Table 2: Overview of notation.

term	description
\mathcal{D}	The space of all possible datasets
d	An instantiation of a dataset
d_{tr}	The instances from a given dataset based upon which a given hypothesis h is trained
\mathcal{X}	The space of all possible input values for a given dataset d
\mathcal{Y}	The space of all possible labels of a given dataset d
H	The space that a given model or hypothesis h can take
h	Model or hypothesis h , induced based on a given train set d_{tr}
a	An algorithm that given a training set d_{tr} induces a hypothesis h
A	Set of all possible learners under consideration
\mathcal{R}_{out}	The theoretical performance of a hypothesis under the true distribution of the data (note that this true distribution is typically unknown)
\mathcal{R}_{in}	The empirical risk of a hypothesis under some sample d (i.e., a dataset) from the true distribution
$\mathcal{C}(a, n)$	true mean performance of learner a when trained on n samples (related to observation learning curve)
$\mathcal{C}(a, n, t)$	true mean performance of learner a when trained on n samples with t iterations (related to iteration learning curve)
$f(a, b)$	The performance of a given learner a trained on a dataset of sample size b (in case of observation curve) or iteration b (in case of iteration curve). In contrast to \mathcal{C} , $f(a, b)$ is a <i>random variable</i> with \mathcal{C} as mean value
$\mu_{a,b}$	The mean of $f(a, b)$
$\sigma_{a,b}^2$	The variance of $f(a, b)$
O	Set of performance observations from anchors of historic learning curves, possibly for different learners, e.g., $O = \{(a_1, b_1), (a_2, b_2), (a_3, b_3), \dots\}$, where a_i are learners and b_j are budgets.
$\hat{f}(a, b)$	The performance estimated by a learning curve model for learner a at budget b . Maybe a point, range, or distributional estimate.
$\hat{f}_a(b)$	The performance estimated by a learning curve model for learner a at budget b if the model does not generalize across learners. Maybe a point, range, or distributional estimate.
n	anchor size, indicating the size of a subsample of the dataset
t	number of iterations, e.g., in the case of neural networks, the number of epochs
b	generic symbol for an anchor, stands either for n or t , depending on the context.
b_{sat}	The anchor size at which the performance of a learner saturates
p_{sat}	The performance of the learner at the saturation point
b_{sat}^u	The anchor size at which the utility (a performance measure divided by a cost measure) is maximized
b_{ref}	Used in the framework for the binary question as a threshold on the budget
p_{ref}	Used in the framework for the binary question as a threshold on the performance
θ	The parameters of a parametric function, for example, the parameters of the IPL-model are (α, β, γ)

Declarations

Funding. Felix Mohr was supported through the project ING-312-2023 from Universidad de La Sabana.

Conflicts of interest. Not Applicable

Ethics approval. Not Applicable

Consent to participate. Not Applicable

Consent for publication. Not Applicable

Availability of data and material. Not Applicable

Code availability. Not Applicable

Authors' contributions. Both authors were involved in the redaction of all parts of the document. F.M. had the lead in all sections; J.N.v.R. contributed to all sections by actively writing and critically reviewing the text.

References

- Adriaensen S, Rakotoarison H, Müller S, et al (2023) Efficient bayesian learning curve extrapolation using prior-data fitted networks. In: Advances in Neural Information Processing Systems 36, pp 19,858–19,886
- Alwosheel A, van Cranenburgh S, Chorus CG (2018) Is your dataset big enough? sample size requirements when using artificial neural networks for discrete choice analysis. Journal of choice modelling 28:167–182
- Amari S, Murata N (1993) Statistical theory of learning curves under entropic loss criterion. Neural Computation 5(1):140–153
- Baker B, Gupta O, Raskar R, et al (2018) Accelerating neural architecture search using performance prediction. In: 6th International Conference on Learning Representations, ICLR’18
- Beleites C, Neugebauer U, Bocklitz T, et al (2013) Sample size planning for classification models. Analytica chimica acta 760:25–33
- Bifet A, Gavaldà R, Holmes G, et al (2018) Machine learning for data streams: with practical examples in MOA. MIT press
- Bishop C (1995) Regularization and complexity control in feed-forward networks. In: Proceedings International Conference on Artificial Neural Networks ICANN’95, pp 141–148
- Boonyanunta N, Zeephongsekul P (2004) Predicting the relationship between the size of training sample and the predictive power of classifiers. In: Knowledge-Based Intelligent Information and Engineering Systems, 8th International Conference, KES 2004. pp 529–535

- Bornschein J, Visin F, Osindero S (2020) Small data, big decisions: Model selection in the small-data regime. In: Proceedings of the 37th International Conference on Machine Learning. pp 1035–1044
- Van den Bosch A (2004) Wrapped progressive sampling search for optimizing learning algorithm parameters. In: Proceedings of the 16th Belgian-Dutch Conference on Artificial Intelligence, pp 219–226
- Brazdil P, van Rijn JN, Soares C, et al (2022) Metalearning: Applications to Automated Machine Learning and Data Mining, 2nd edn. Springer
- Cardona-Escobar AF, Giraldo-Forero AF, Castro-Ospina AE, et al (2017) Efficient hyperparameter optimization in convolutional neural networks by learning curves prediction. In: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. pp 143–151
- Chandrashekaran A, Lane IR (2017) Speeding up hyper-parameter optimization by extrapolation of learning curves using previous builds. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017. pp 477–492
- Cho J, Lee K, Shin E, et al (2015) How much data is needed to train a medical image deep learning system to achieve necessary high accuracy? CoRR abs/1511.06348. <https://arxiv.org/abs/1511.06348>
- Cortes C, Jackel LD, Solla SA, et al (1993) Learning curves: Asymptotic values and rate of convergence. In: Advances in Neural Information Processing Systems 6. pp 327–334
- Cortes C, Jackel LD, Chiang W (1994) Limits in learning machine accuracy imposed by data quality. In: Advances in Neural Information Processing Systems 7. pp 239–246
- da Costa FG, Rios RA, de Mello RF (2016) Using dynamical systems tools to detect concept drift in data streams. Expert Systems with Applications 60:39–50
- Domhan T, Springenberg JT, Hutter F (2015) Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015. pp 3460–3468
- Domingos P, Hulten G (2000) Mining High-Speed Data Streams. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 71–80

- Dong X, Yang Y (2020) Nas-bench-201: Extending the scope of reproducible neural architecture search. In: 8th International Conference on Learning Representations, ICLR 2020
- Egele R, Mohr F, Viering T, et al (2024) The unreasonable effectiveness of early discarding after one epoch in neural network hyperparameter optimization. *Neurocomputing* 597:127,964
- Eggensperger K, Lindauer M, Hoos HH, et al (2018) Efficient benchmarking of algorithm configurators via model-based surrogates. *Machine Learning* 107(1):15–41
- Eggensperger K, Müller P, Mallik N, et al (2021) HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks
- Figueredo RL, Zeng-Treitler Q, Kandula S, et al (2012) Predicting sample size required for classification performance. *BMC Medical Informatics Decis Mak* 12:8
- Fine T, Mukherjee S (1999) Parameter convergence and learning curves for neural networks. *Neural Comput* 11(3):747–769
- Forman G, Cohen I (2004) Learning from little: Comparison of classifiers given little training. In: Knowledge Discovery in Databases: PKDD 2004, 8th European Conference on Principles and Practice of Knowledge Discovery in Databases. pp 161–172
- Frey LJ, Fisher DH (1999) Modeling decision tree performance with the power law. In: Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics, AISTATS 1999
- Fürnkranz J, Petrik J (2001) An evaluation of landmarking variants. In: Working Notes of the ECML/PKDD 2000 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning, pp 57–68
- Gkioxari G, Toshev A, Jaitly N (2016) Chained predictions using convolutional neural networks. In: Leibe B, Matas J, Sebe N, et al (eds) Computer Vision - ECCV 2016 - 14th European Conference. pp 728–743
- Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press, <http://www.deeplearningbook.org>
- Gu B, Hu F, Liu H (2001) Modelling classification performance for large data sets. In: Advances in Web-Age Information Management, Second International Conference, WAIM 2001. pp 317–328

Hess KR, Wei C (2010) Learning curves in classification with microarray data. *Seminars in oncology* 37(1):65–68

Hollmann N, Müller S, Eggensperger K, et al (2023) TabPFN: A transformer that solves small tabular classification problems in a second. In: The Eleventh International Conference on Learning Representations, ICLR 2023

Hughes GF (1968) On the mean accuracy of statistical pattern recognizers. *IEEE Trans Inf Theory* 14(1):55–63

Hüllermeier E, Waegeman W (2021) Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning* 110(3):457–506

Jamieson KG, Talwalkar A (2016) Non-stochastic best arm identification and hyperparameter optimization. In: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016. pp 240–248

John GH, Langley P (1996) Static versus dynamic sampling for data mining. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). pp 367–370

Kielhofer L, Mohr F, van Rijn JN (2024) Learning curve extrapolation methods across extrapolation settings. In: Advances in Intelligent Data Analysis XXII. pp 145–157

Klein A, Falkner S, Bartels S, et al (2017a) Fast bayesian optimization of machine learning hyperparameters on large datasets. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017. pp 528–536

Klein A, Falkner S, Springenberg JT, et al (2017b) Learning curve prediction with bayesian neural networks. In: 5th International Conference on Learning Representations, ICLR'17

Kolachina P, Cancedda N, Dymetman M, et al (2012) Prediction of learning curves in machine translation. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference. pp 22–30

Koshute P, Zook J, McCulloh I (2021) Recommending training set sizes for classification. CoRR abs/2102.09382. <https://arxiv.org/abs/2102.09382>

Last M (2007) Predicting and optimizing classifier utility with the power law. In: Workshops Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007). pp 219–224

- Last M (2009) Improving data mining utility with projective sampling. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp 487–496
- Leite R, Brazdil P (2003) Improving progressive sampling via meta-learning. In: Progress in Artificial Intelligence, 11th Portuguese Conference on Artificial Intelligence, EPIA 2003. pp 313–323
- Leite R, Brazdil P (2004) Improving progressive sampling via meta-learning on learning curves. In: Machine Learning: ECML 2004, 15th European Conference on Machine Learning. pp 250–261
- Leite R, Brazdil P (2005) Predicting relative performance of classifiers from samples. In: Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005). pp 497–503
- Leite R, Brazdil P (2007) An iterative process for building learning curves and predicting relative performance of classifiers. In: Progress in Artificial Intelligence, 13th Portuguese Conference on Artificial Intelligence, EPIA 2007. pp 87–98
- Leite R, Brazdil P (2008) 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
- Leite R, Brazdil P (2010) Active testing strategy to predict the best classification algorithm via sampling and metalearning. In: ECAI 2010 - 19th European Conference on Artificial Intelligence. pp 309–314
- Li L, Jamieson KG, DeSalvo G, et al (2017) Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research* 18:185:1–185:52
- Long D, Zhang S, Zhang Y (2020) Performance prediction based on neural architecture features. *Cognitive Computation and Systems* 2(2):80–83
- Loog M, Duin RP (2012) The dipping phenomenon. In: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), pp 310–317
- Loog M, Viering TJ, Mey A (2019) Minimizers of the empirical risk and risk monotonicity. In: Advances in Neural Information Processing Systems 32, pp 7476–7485
- Mao Z, Jung T, Lin C, et al (2016) Predicting EEG sample size required for classification calibration. In: Foundations of Augmented Cognition: Neuroergonomics and Operational Neuroscience - 10th International Conference, AC

2016. pp 57–68

Meek C, Thiesson B, Heckerman D (2002) The learning-curve sampling method applied to model-based clustering. *Journal of Machine Learning Research* 2:397–418

Mhammedi Z, Husain H (2021) Risk-monotonicity in statistical learning. In: *Advances in Neural Information Processing Systems* 34, pp 10,732–10,744

Mohr F, van Rijn JN (2021) Towards model selection using learning curve cross-validation. In: 8th ICML Workshop on Automated Machine Learning (AutoML)

Mohr F, van Rijn JN (2023) Fast and informative model selection using learning curve cross-validation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45(8):9669–9680

Mohr F, Viering TJ, Loog M, et al (2022) LCDB 1.0: An extensive learning curves database for classification tasks. In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD*. pp 3–19

Mørch NJS, Hansen LK, Strother SC, et al (1997) Nonlinear versus linear models in functional neuroimaging: Learning curves and generalization crossover. In: *Information Processing in Medical Imaging, 15th International Conference, IPMI'97*. pp 259–270

Mukherjee S, Tamayo P, Rogers S, et al (2003) Estimating dataset size requirements for classifying DNA microarray data. *Journal of Computational Biology* 10(2):119–142

Murata N, Yoshizawa S, Amari S (1992) Learning curves, model selection and complexity of neural networks. In: *Advances in Neural Information Processing Systems* 5. pp 607–614

Nakkiran P, Kaplun G, Bansal Y, et al (2020) Deep double descent: Where bigger models and more data hurt. In: 8th International Conference on Learning Representations, ICLR'20

Nakkiran P, Venkat P, Kakade SM, et al (2021) Optimal regularization can mitigate double descent. In: 9th International Conference on Learning Representations, ICLR 2021

Ng AY, Jordan MI (2001) On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In: *Advances in Neural Information Processing Systems* 14. pp 841–848

- Ng W, Dash M (2006) An evaluation of progressive sampling for imbalanced data sets. In: Workshops Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006). pp 657–661
- Oyedare T, Park JJ (2019) Estimating the required training dataset size for transmitter classification using deep learning. In: 2019 IEEE International Symposium on Dynamic Spectrum Access Networks, DySPAN 2019. pp 1–10
- Perlich C, Provost FJ, Simonoff JS (2003) Tree induction vs. logistic regression: A learning-curve analysis. *Journal of Machine Learning Research* 4:211–255
- Petrak J (2000) Fast subsampling performance estimates for classification algorithm selection. In: Proceedings of the ECML-00 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination, pp 3–14
- Pfisterer F, Schneider L, Moosbauer J, et al (2022) YAHPO gym - an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In: International Conference on Automated Machine Learning, AutoML. pp 3/1–39
- Provost FJ, Jensen DD, Oates T (1999) Efficient progressive sampling. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp 23–32
- Richter AN, Khoshgoftaar TM (2019) Approximating learning curves for imbalanced big data with limited labels. In: 31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019. pp 237–242
- van Rijn JN, Abdulrahman SM, Brazdil P, et al (2015) Fast algorithm selection using learning curves. In: Advances in Intelligent Data Analysis XIV. pp 298–309
- Ruhkopf T, Mohan A, Deng D, et al (2023) Masif: Meta-learned algorithm selection using implicit fidelity information. *Transactions on Machine Learning Research* 2023
- Sabharwal A, Samulowitz H, Tesauro G (2016) Selecting near-optimal learners via incremental data allocation. In: Proceedings of the AAAI Conference on Artificial Intelligence
- Sarkar A, Guo J, Siegmund N, et al (2015) Cost-efficient sampling for performance prediction of configurable systems (T). In: 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015. pp 342–352

- Settles B (2009) Active learning literature survey. Tech. rep., University of Wisconsin
- Seung HS, Sompolinsky H, Tishby N (1992) Statistical mechanics of learning from examples. *Physical Review A* 45(8):6056
- Siems J, Zimmer L, Zela A, et al (2020) Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. CoRR abs/2008.09777. <https://arxiv.org/abs/2008.09777>
- Singh S (2005) Modeling performance of different classification methods: deviation from the power law. Project Report, Department of Computer Science, Vanderbilt University, USA
- Strang B, van der Putten P, van Rijn JN, et al (2018) Don't rule out simple models prematurely: A large scale benchmark comparing linear and non-linear classifiers in openml. In: Advances in Intelligent Data Analysis XVII. pp 303–315
- Swersky K, Snoek J, Adams RP (2014) Freeze-thaw bayesian optimization. CoRR abs/1406.3896. <https://arxiv.org/abs/1406.3896>
- Tomanek K (2010) Resource-aware annotation through active learning. PhD thesis, Dortmund University of Technology
- Vallet F, Cailton JG, Refregier P (1989) Linear and nonlinear extension of the pseudo-inverse solution for learning boolean functions. *EPL (Europhysics Letters)* 9(4):315
- Viering TJ, Loog M (2023) The shape of learning curves: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45(6):7799–7819
- Viering TJ, Mey A, Loog M (2020) Making learners (more) monotone. In: Advances in Intelligent Data Analysis XVIII. pp 535–547
- Waltz M, Fu K (1965) A heuristic approach to reinforcement learning control systems. *IEEE Transactions on Automatic Control* 10(4):390–398
- Wang X, Chen Y, Zhu W (2022) A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44(9):4555–4576
- Weiss GM, Provost FJ (2003) Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research* 19:315–354
- Weiss GM, Tian Y (2006) Maximizing classifier utility when training data is costly. *SIGKDD Explorations* 8(2):31–38

Weiss GM, Tian Y (2008) Maximizing classifier utility when there are data acquisition and modeling costs. *Data Mining and Knowledge Discovery* 17(2):253–282

White C, Safari M, Sukthanker R, et al (2023) Neural architecture search: Insights from 1000 papers. CoRR abs/2301.08727. <https://arxiv.org/abs/2301.08727>

Wistuba M, Pedapati T (2019) Inductive transfer for neural architecture optimization. CoRR abs/1903.03536. <https://arxiv.org/abs/1903.03536>

Zeng X, Luo G (2017) Progressive sampling-based bayesian optimization for efficient and automatic machine learning model selection. *Health Information Science and Systems* 5(1):2