Universiteit Leiden
The Netherlands

Massively collaborative machine learning

Rijn, J.N. van

3 Meta-Learning

Machine Learners have been very successful at integrating data-driven methods in many other domains and sciences: Chemist model the activity of chemical compounds by means of Machine Learning [81], popular media companies use knowledge obtained from previous data to recommend new content to their subscribers, and modern cars use Machine Learning techniques to take away the burden of parking.

However, when it comes to Machine Learning itself, decisions are seldom made by what previous gathered data has learned. When presented with a new problem, often a solution is chosen by trial and error. Typically, a small set of algorithms is selected by intuition, and the best of these is selected to solve the problem. This trial and error is sometimes called 'the black art of Machine Learning' [38], implying that a more scientific approach is desirable.

A huge challenge lies in solving Machine Learning problems in a data-driven way. This scientific challenge is called *meta-learning*; the research field that aims to learn from previous applications and experiments.

3.1 Introduction

The field of meta-learning has attracted quite some attention, and quite some problems and challenges in various directions have been addressed. Although it is impossible to capture the whole field in one single definition, we will consider the definition by Vilalta et al. [159]: "The field of meta-learning has as one of its primary goals the understanding of the interaction between the mechanism of learning and the concrete contexts in which that mechanism is applicable." In this definition, the 'mechanisms of learning' are the algorithms (that build models) and the 'contexts in

which that mechanism is applicable' are the Machine Learning tasks and datasets. Basically, we want to obtain knowledge (or learn) which algorithm and parameter setting should be used on what kind of data.

This problem can be viewed from many different perspectives. Most notable, the *algorithm selection problem*, is defined as: given a dataset, which algorithm and parameters will obtain maximal performance according to a specified measure [119]. As the amount of algorithm and parameter combinations is infinite, this in itself is already a hard but important problem. Considering common applications of Machine Learning, e.g., in healthcare and epidemiology, performing slightly better on a given task can already result in making the difference for human lives.

In many realistic settings, the algorithm selection problem can also be seen as a *search problem*. A model, algorithm and parameter settings are recommended, and these are being tested (either in production or using cross-validation). For example, a company modelling customer behaviour, could already apply the recommended model in their production environment. However, they can still continuously test other models to find one that improves upon the original selected model. This process is repeated until a satisfactory model has been found.

Sometimes, the model is fixed a priori because of, e.g., empirical performance evidence, or interpretability requirements. In this case, the main challenge is to select the appropriate hyperparameters. Hence, this task is called *hyperparameter optimization* (e.g., [8, 75, 89]). Many techniques from the optimization literature can be used for this, e.g., Particle Swarm Optimization, Evolutionary Algorithms or Bayesian optimization.

Rather than looking at algorithm performance, we can also look at dataset properties. Where meta-learning is often focussed on categorizing datasets as a whole entity, Smith et al. [140] focusses on individual instances, attempting to categorize which are often misclassified, why this happened and how they contribute to data set complexity. This research potentially improves the learning process and can guide the development of learning algorithms.

The remainder of this chapter is organised as follows. We discuss three general approaches to algorithm selection. Section 3.2 approaches this from the learning paradigm; Section 3.3 approaches this from the search paradigm; Section 3.4 introduces the notion of ensembles. Next, we introduce some important aspects to meta-learning. Section 3.5 discusses the 'Law of Conservation for Generalization Performance', which is sometimes erroneously cited to dismiss meta-analysis. Section 3.6 discusses some analytically obtained knowledge about models, which is a very general form of meta-knowledge. Section 3.7 examines the bias variance trade-off, and the dilemma that comes with choosing between over-fitting and under-fitting. Section 3.8 concludes with a discussion.

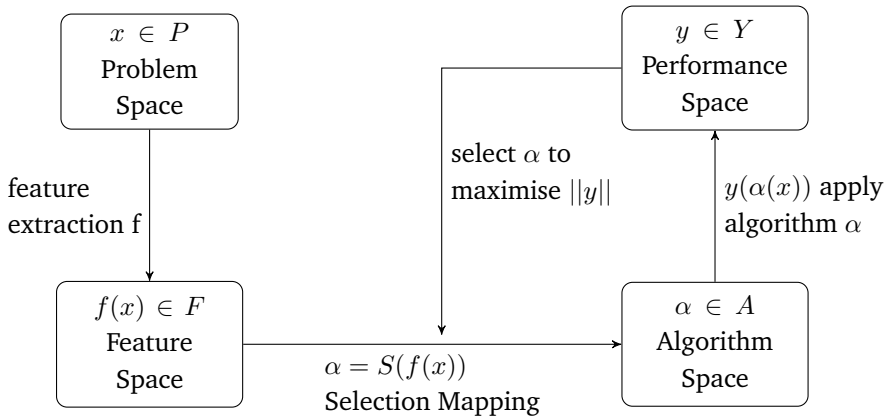

Figure 3.1: The Algorithm Selection Framework, taken from [141].

3.2 Learning Approach

The algorithm selection problem occurs in many other fields besides Machine Learning. Many optimization problems can be solved with a wide range of algorithms and parameters, and therefore there is room for algorithm selection. For example, satisfiability problems, travelling salesman problems and vehicle routing problems.

The algorithm selection framework, formalised by Rice [119], addresses this. The framework is illustrated in Figure 3.1. According to this definition, the problem space P consists of all problems (tasks) from a certain domain. Each problem $x \in P$ can be mapped by means of some feature extraction procedure $f(x)$ into the feature space F. The feature space F contains measurable characteristics calculated upon these problems, e.g., the number of instances or the number of attributes. We call these *meta-features*. The algorithm space A is the set of all considered algorithms that can be used to solve these tasks. And finally, the performance space Y represents the mapping of these algorithms to a set of performance measures. The task is for any given $x \in P$, to select the algorithm $\alpha \in A$ that maximizes a predefined performance measure $y \in Y$.

Essentially, this itself is a learning problem, and can be solved using conventional Machine Learning techniques. Often, Random Forests are used [144]. Like with any learning problem, there are instances, represented by features that have a certain class feature. In this case, the best performing algorithm is the class attribute. This task can also be casted as a ranking or a regression problem. In the case of the ranking problem, the goal is to order the classifiers by their expected performance; in the case of the regression problem, the goal is to predict the expected performance for a set of

given algorithms.

Indeed, by approaching the algorithm selection problem as a learning problem, all Machine Learning algorithms can be used as meta-learner and solve the algorithm selection problem. Various solutions solving this problem in novel ways have been proposed (see, e.g., [4, 27, 87, 144]).

Some non-trivial considerations remain. Most importantly, as with any modelling task, the set of (meta-)features determines the quality of the solution. These need to be chosen and constructed appropriately. Furthermore, the performance space determines on what performance criteria the algorithm should be selected.

3.2.1 Feature space

The performance of a meta-learning solution typically depends on the quality of the meta-features. Typical meta-features are often categorized as either simple, statistical, information theoretic, algorithm/model-based or landmarkers.

The simple meta-features can all be calculated by one single pass over all instances and describe the data set in an aggregated manner, e.g., number of instances, number of attributes and number of classes [20]. The statistical meta-features are calculated by considering a statistical concept (e.g., standard deviation, skewness or kurtosis), calculate this for all numeric attributes and taking the mean of this. This leads to, e.g., the mean standard deviation of numeric attributes. Other statistical aggregation methods can also be used instead of the mean, e.g., the minimum, the maximum, or the median. Likewise, the information theoretic meta-features are calculated by considering an information theoretic concept (e.g., mutual information or attribute entropy), calculate this for all nominal attributes and taking the mean of this. This leads to, e.g., mean mutual information. Again, other statistical aggregation methods can be used as well. By aggregating the statistical and information theoretic concepts, information is lost by definition. A main challenge lies in finding a way of preserving this information. Sometimes the meta-data is built upon datasets with the exact same features. In that case, no aggregation over the meta-features is needed; these can be calculated and used for every individual features, without losing any information [164].

Landmarkers are performance evaluations of fast algorithms on a dataset [108]. Sometimes, knowledge about how simple classifiers perform on a dataset yields information on the performance of the more complex and time-consuming algorithms. However, these procedures should be used with great care, as the time for calculating the meta-features should not exceed the time of bluntly running all algorithms on the dataset.

Taking this idea one step further leads to model-based features [104]. Again, a

simple model is built upon the data, most commonly, decision trees. Topological properties from this model can be used as meta-features, e.g., shape, number of leafs, or maximum tree depth.

Sun and Pfahringer [144] propose pairwise meta-rules. These are simple decision rules that determine for each pair of algorithms, which will work best under which conditions. This type of meta-feature assumes that plain information from landmarkers is not necessarily represented best in its numeric form, and transforms this information to a binary attribute, at the cost of additional computation time.

It has proven hard to come up with an appropriate set of meta-features. Therefore, Provost et al. [113] proposes Partial Learning Curves as an intuitive way of mapping a problem into the feature space. A learning curve is an ordered set of performance scores of a classifier on data samples of increasing size. Intuitively, we have now information on how the actual classifiers that we are interested in work on the actual dataset that we are interested in. Various methods have been proposed that exploit partial learning curves to solve the algorithm selection problem [86, 87, 127].

Pinto et al. [110] propose a framework that enables the systematic generation of meta-features specific to a certain domain. The framework detects what kind of meta-features can lead to the generation of new meta-features. For example, when using a decision tree landmarker, this can trigger the generation of model-based meta-features extracted from the decision tree. Their results indicate that sets of systematically generated meta-features are more informative and contain more predictive power than ad-hoc selected meta-features.

3.2.2 Performance space

Commonly, solutions to the algorithm selection problem focus on finding an algorithm that maximizes predictive accuracy. This makes sense for a variety of theoretical and pragmatic reasons; it objectively orders the full set of available algorithms, is easy to establish (e.g., by means of cross-validation) and is what we are commonly interested in [59]. However, as there are many tasks where almost all algorithms perform well in terms of accuracy [69], it makes sense to also consider other ways of selection criteria for algorithms. Some options for this are computational complexity, compactness of the resulting model or comprehensibility of the resulting model [59]. Sometimes it is sensible to make a trade-off between accuracy and run-time [21, 127]. In some problem domains, the balance between the classes is skewed. Although in those cases it might be easy to obtain a high accuracy by predicting the majority class, it is more interesting to have a model that performs well on the other class(es). Measures such as area under the ROC curve, precision and f-measure are decent options for those problems.

Furthermore, it is debatable whether the single best model should be recommended. Alternatively, a statistical test could be performed, and all algorithms that perform statistically equivalent to the classifier obtaining the highest score are considered good recommendations. Intuitively, this makes sense: if there is no statistical evidence that there is a difference between the performance of the algorithms, then there is no strong evidence to prefer one over the other. In many modern applications, it is not enough to predict a single algorithm. Rather than predicting one single algorithm, a ranking is produced, giving alternatives if the first advised algorithm does not perform adequately. The algorithm selection problem could even be seen as a regression problem: for each algorithm $\alpha \in A$ an estimated performance should be predicted, and the algorithm with the highest estimated score can be selected.

3.3 Search Approach

Alternatively, the algorithm selection problem can be cast as a search problem, which can be solved by many techniques from the field of black-box optimization. Typically, an optimization algorithm recommends an algorithm and parameter setting, and it is then tested using some evaluation procedure, e.g., cross-validation. It continues on recommending these procedures, until the budget runs out. Then the best performing algorithm and parameter setting combination is selected.

In some cases the task is to find the best algorithm and parameter setting, in other cases the algorithm is fixed a priori, and the task is to find the optimal parameter settings. This task is called *hyperparameter optimization*.

A commonly used search strategy for hyperparameter optimization is *grid search*, which bluntly tries all possible parameter settings. As many parameters accept continues values, for most algorithms there are infinitely many parameter settings. Therefore grid search requires the intervention of a human expert, who selects sensible parameters, ranges and discretizations. Grid search has no natural way of dealing with a budget.

To overcome these limitations, *random search* could be used as an alternative. As the name suggests, it randomly tries some parameter settings from all possibilities. This way, there is no real need for the human expert to select parameters, ranges and discretizations (although it can still benefit from the input of a human expert). Furthermore, it naturally deals with a budget, as it can just stop whenever the budget runs out. The best parameter settings so far will be selected. Random search often performs better than grid search. When exploring the same parameter space it finds an acceptable setting much faster, and when using a fixed budget it goes beyond the selected parameters and ranges that grid search is restricted to [8].

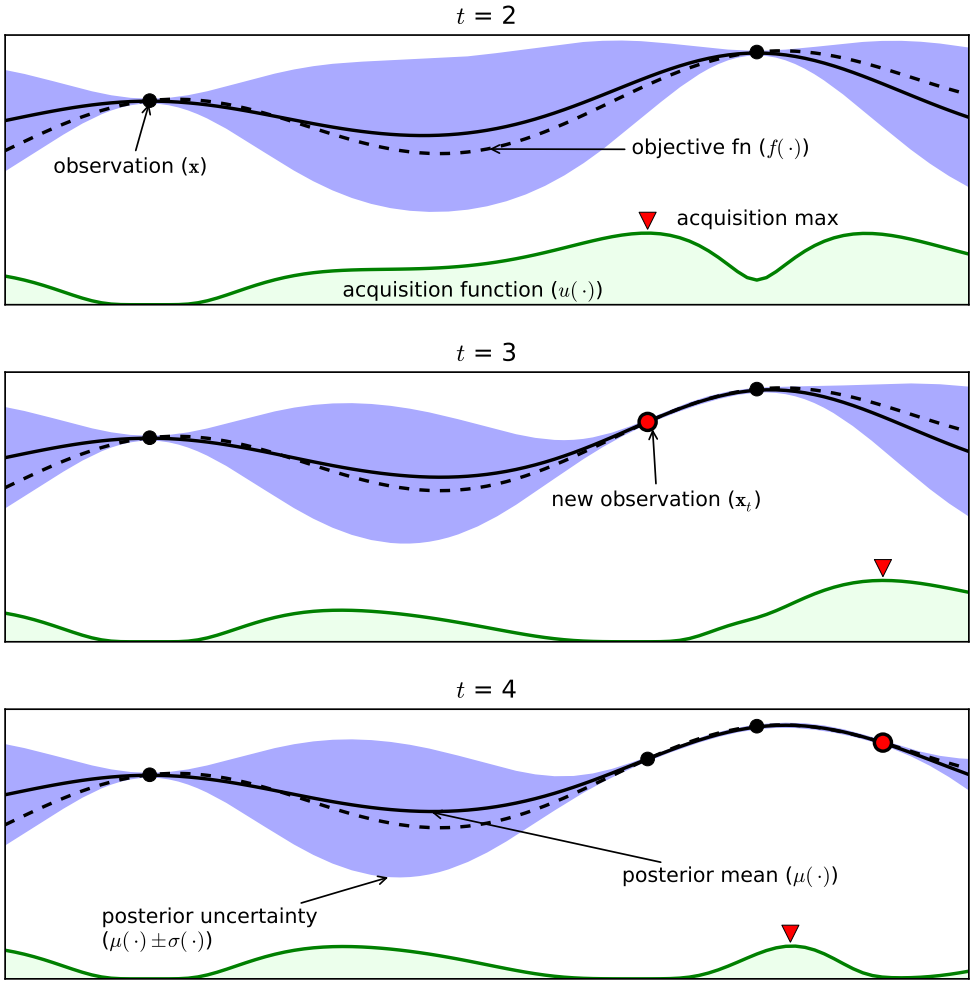

Figure 3.2: Example of Bayesian Optimization on one variable. The dashed line represents the objective function, which we only know partly (at the black dots); the striped line represents the posterior mean, and the purple area the posterior uncertainty; the acquisition function is plotted in green. Image taken from [26].

Both grid search and random search are *static search methods*, information obtained from results of earlier test is ignored. Opposite to this, *dynamic search methods* attempt to use this information. For example, when a certain parameter setting obtains good results, it is plausible that similar parameter settings will also obtain good results and possibly improve on the current best result.

Bayesian Optimization is a strategy that attempts to exploit knowledge obtained from earlier tests [26]. The parameter space is often modelled as a *Gaussian process*; it assumes that the underlying function is smooth, i.e., small fluctuations in the parameter settings will lead to only small fluctuations in the result. A so-called *acquisition function* determines which parameter setting is evaluated next based on high uncertainty and high potential. Figure 3.2 illustrates this in an example, in which one parameter is optimized. At each point t, it selects a new parameter setting that is being evaluated. Sequential Model-based Bayesian Optimization extends this idea, also selecting among various learning algorithms [75]. Notably, Auto-Weka applies this on Machine Learning, automatically searching for a good algorithm and parameter setting among all algorithms from the toolbox WEKA [147].

The search problem can also be modelled as a *Multi-armed Bandit problem* [68, 89], named appropriately after a gambler who's aim is to select a slot machine that gives him maximal reward. Typically there is a set of *arms*, each representing an algorithm with parameter configuration. Pulling an arm is associated with a certain cost (training a model on an amount of data) and reward (the performance of this model). Pulling a certain arm multiple times corresponds to training that model on an increasing amount of data, gaining a higher confidence in the measured performance of this model. Two notable algorithms are Successive Halving and Hyperband [89]. These start with testing a large number of algorithms on a small amount of data; badly performing algorithms are eliminated and good performing algorithms are tested with more data. This works very well in practise. Additionally, these algorithms come with theoretical guarantees about the maximum *regret*, i.e., the difference between the recommended algorithm and parameter setting and the absolute best one.

Despite these sophisticated techniques and theoretical guarantees, the simplicity of Random Search still proves to be a strong baseline. Given twice the budget, it often outperforms guided search schema's [9].

3.3.1 Combining Search and Learning

Various strategies have been proposed that combine the thoroughness of the search strategies with the knowledge-based approach of meta-learning.

Active Testing combines grid search with meta-knowledge [2, 88], intelligently selecting the order in which the algorithm and parameter combinations are tried. It aims to minimize the number of algorithms that need to be evaluated before an adequate model has been built. It assumes that there is a meta-dataset containing information on how the algorithms and parameter settings under consideration performed on other problems. Furthermore, it assumes that already one algorithm is selected as the most promising to try first. This could be the one that performed best on historic

data, an algorithm recommended by another meta-learning system, or even one that was randomly selected. This algorithm is called the *current best*. From then on, the algorithm that outperforms the current best algorithm on most historic datasets that seem similar to the current dataset, is tested next. For each new algorithm to try, it only focusses on historic datasets on which the new algorithm outperforms the current best. This is inspired by the idea that we are interested in volatile algorithms. Naturally, if the newly tried algorithm turns out to be better than the current best, from that moment on it is the current best. This process is repeated until an appropriate algorithm has been selected or a predefined budget (e.g., run time) runs out. Chapter 6 details on this method.

Alternatively, meta-learning can be used to initialize complex search methods. It has been known that Sequential Model-based Bayesian Optimization converges fast when its initial points yield already good performance. Hence, meta-learning can be natively used to find promising parameter settings on similar datasets [43]. These can be used as initial evaluations for the Sequential Model-based Bayesian Optimization, leading to faster convergence.

3.4 Ensembles

Another approach to select the best model, is to combine multiple models in an *ensemble* of classifiers. Ensemble techniques train multiple classifiers (also called *members*) on a set of weighted training examples; these weights can vary for different classifiers. In order to classify test examples, all individual models produce a prediction, also called a *vote*, and the final prediction is made according to a predefined voting policy. For example, the class with the most votes is selected. Dietterich [37] identifies three reasons why ensembles work better than individual models.

- **Statistical**. When there is insufficient training data, there will be multiple models that fit the training set well, but have various (unknown) performance on the test set. Combining multiple models spreads the risk of a misclassification among multiple models.

- **Computational**. Even when there is sufficient data, the learning algorithm that induces the model might get stuck in a local optimum. Running the learning algorithm multiple times from various starting points may give a better performance on the unknown test set.

- **Representational**. In many Machine Learning applications, the true concept that is being modelled can not be represented by a given algorithm. For example, Logistic Regression can not represent the XOR-function, because it is

not linearly separable (see Chapter 2.4.5). However, an ensemble of multiple Logistic Regression models is able to perfectly represent it.

Condorcet's jury theorem [83] gives theoretical evidence that the error rate of an ensemble in the limit goes to the Bayesian optimum (i.e., the maximum obtainable accuracy) if two conditions are met. First, the individual models must do better than random guessing. Second, the individual models must be diverse, i.e., their errors should not be correlated. If these conditions are met, increasing the ensemble size decreases the amount of misclassifications [64]. Indeed, if the models do worse than random guessing, increasing the ensemble size also increases the amount of misclassifications.

Basically, two approaches of model combination exist [22]. The first one exploits variability in the data, and trains similar models on different subsamples of the data. The second one exploits variability among models, and trains fundamentally different models on the same data. *Bagging* [23] exploits the instability of classifiers by training them on different subsamples called *bootstrap replicates*. A bootstrap replicate is a set of examples randomly drawn with replacement, to match the size of the original training set. Some examples will occur multiple times, some will not occur in the bootstrap replicate. Bagging works particularly good with *unstable* algorithms, were small fluctuations in the training set lead to dissimilar models. Bagging reduces the variance error of a model, and only slightly affects the bias error. Algorithms that have a high variance error typically perform much better when used in a Bagging setting, at the cost of losing interpretability.

Boosting [135] is a technique that corrects the bias of weak learners. A *strong learner* is one that produces a highly accurate model; a *weak learner* is one whose models perform slightly better than random guessing [149]. In his seminal paper, Schapire [135] shows weak learners and strong learners are equivalent; he presents an algorithm that combines various weak learners that combined perform as a strong learner. Boosting sequentially trains multiple classifiers, in which more weight is given to examples that where misclassified by earlier classifiers. It decreases both bias and variance error. Some common forms of boosting are Adaptive Boosting [45], Logistic Boosting [48] and Gradient Boosting [51].

Stacking [163] combines heterogeneous models in the classical batch setting. It trains multiple models on the training data. All members output a prediction, and a meta-learner makes a final decision based on these predictions. *Cascade Generalization* [53] imposes an order to the ensemble members. Each member makes a prediction also based on the prediction of the previous members. Empirical evidence suggests that Cascade Generalization performs better than vanilla Stacking. Caruana et al. [30] propose a hill-climbing method to select an appropriate set of base-learners from a large library of models.

3.5 Conservation for Generalization Performance

The 'Law of Conservation for Generalization Performance' (also known as the No Free Lunch Theorem), states that when taken across all learning tasks, all learning algorithms perform equally well [134]. Basically, it states that for each algorithm there are datasets on which it performs well, and there are datasets on which it performs badly; each algorithm makes its own assumptions about the data.

This theorem can be illustrated by means of the following example. Consider a binary dataset d on which a deterministic algorithm α obtains a certain predictive accuracy $y(\alpha, d)$ on a test set of unseen examples. In the universe of all imaginable datasets, there exists a dataset $\hat{d}$ that has the exact same training instances, yet all class labels of the unseen test set are exactly opposite, leading to a predictive accuracy of $y(\alpha, \hat{d})$. It is easy to see that in this example $y(\alpha, d) + y(\alpha, \hat{d}) = 1$. A generalization leads to the conclusion that taken over all existing datasets, the performance of all deterministic algorithms is exactly the same.

It might seem that this theoretical result subverts the purpose of meta-learning. If all algorithms perform equally, building a meta-algorithm that overcomes this limitation would be paradoxical. However, Giraud-Carrier and Provost [60] point out that under a reasonable assumption, the 'Law of Conservation for Generalization Performance' is irrelevant to Machine Learning research.

They define *the weak assumption of Machine Learning* to be that the process that presents us with learning problems induces a non-uniform probability distribution over the possible functions. In other words, among all possible learning tasks, some are more probable to occur than others. This assumption is widely accepted; without accepting this assumption, all Machine Learning research would be pointless, as there would be no reason to believe that the learned models generalize beyond the dataset. Under this assumption, meta-learning does not violate the law of conservation for generalization performance.

Furthermore, they define the *the strong assumption of Machine Learning* that the probability distribution of these functions is known implicitly or explicitly, at least to a useful approximation. In some cases, even this assumption is reasonable. A good meta-learning system selects algorithms that work well on the tasks that are probable to occur in the problem domain of interest. A meta-learning system based on knowledge from one domain of tasks can not be expected to make accurate recommendations for tasks from another domain.

The most important contribution of the 'Law of Conservation for Generalization Performance' is that it shows that learning comes with assumptions. The stronger the assumptions we make, the more efficient a learning procedure can be. Meta-learning relies on the same assumptions as any other learning procedure does.

Table 3.1: Some characteristics about different models. Taken from [65].

Characteristic	Neural Networks	SVM	Trees	MARS	k-NN, Kernels
Natural handling of mixed data types	-	-	+	+	-
Handling of missing values	-	-	+	+	+
Robustness to outliers in input space	-	-	+	-	+
Insensitive to monotone transformations of inputs	-	-	+	-	-
Computational Scalability (many instances)	-	-	+	+	-
Ability to deal with irrelevant inputs	-	-	+	+	-
Ability to extract linear combinations of features	+	+	-	-	+/-
Interpretability	-	-	+/-	+	-
Predictive Power	+	+	-	+/-	+

3.6 Model Characteristics

Although applications to the algorithm selection problem (Chapter 3.2–3.4) are useful in their own right and lead to various valuable insights, a disadvantage is that they typically focus on a small set of problems, and that it is hard to interpret and generalize the results. On the other side of the spectrum, Hastie et al. [65] composed a set of simple characteristics, describing strong and weak points of models. These are shown in Table 3.1. Most of these models are already described in Chapter 2.4. MARS [49] is a regression model based on *splines*, a mathematical function that consist of various polynomials at different input domains.

Most of the characteristics actually make a lot of sense. For example, it followed already from the model description in Chapter 2.4 that Logistic Regression, Support Vector Machines and Neural Networks do not handle missing values natively, as it would remove one part of the equation. Furthermore, trees and splines are typically considered quite interpretable, even though this is a subjective matter. This kind of simple characteristics already have great value. It summarizes the strong and weak points of various models, and gives domain experts that use machine learning a good overview of what kind of models they should use for their problem.

The mentioned characteristics also have shortcomings. For example, the results

are presented without any discussion or evidence. Although most of the statements seem intuitively correct, there are also some controversies. For example, it is widely considered that decision trees have a high ability to deal with irrelevant inputs. However, experimental results suggest that the truth is a bit more subtle [111]. Also, the low robustness to outliers in input space of Support Vector Machines is debatable.

This table immediately exploits a huge problem in current Machine Learning literature. New models are typically evaluated solely based on predictive power; sometimes interpretability and computational scalability are also taken into account. However, every new method should be evaluated on more than just these criteria. Of course, a method does not have to excel on all of the characteristics. As can be seen, none of the currently characterised models do. Extending this table to additional models, algorithms and characteristics is a very useful form of meta-learning.

3.7 Bias Variance Profile

A common problem of modelling is *over-fitting*, a phenomenon where irrelevant relationships between the data and the class are being encoded. This typically happens when there is too little data, there is too much noise in the data or the model is too complex. When a model performs well on the training set and mediocre on the test set, it is likely that it over-fitted the training data. One way of analysing and understanding this better, is by means of a *bias variance decomposition* [82]. Kohavi et al. [82] define three types of errors, *bias* errors, *variance* errors and *irreducible errors*. The irreducible errors are the errors that can not be avoided by any learner, e.g., when the test set contains instances with the same attribute values, yet different labels. Variance is the tendency to learn random things irrespective of the real function (it hallucinates patterns). This often happens when a model is too complex, and models dependencies that do not exist in the real world; it over-fits the data. Bias is the tendency to consistently learn the same wrong thing. This often happens when a model is too simple, and unable to model the true relationship between the data and the class; it *under-fits* the data. Bias and variance are often resembled by throwing darts at a board (see Figure 3.3). A model that only makes irreducible errors is called a *Bayes-optimal* model.

Many methods attempt to prevent over-fitting. One way of doing so is adding a *regularization component*, penalizing complex models, and therefore favouring simpler models with less room to over-fit. However, by imposing this preference towards simpler models, the solution is exposed to the other type of error, i.e., under-fitting. A notable exception is Bagging, a technique that reduces the risk of over-fitting with slightly increasing the bias (Chapter 3.4). Dealing with bias and variance is essentially

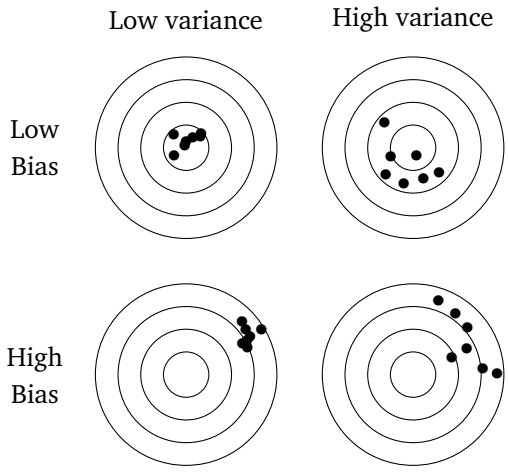

Figure 3.3: Bias and Variance. Image adapted from [38].

about finding a trade-off between the possibility of over-fitting and under-fitting. As additional components are added to the model, the model complexity rises, the bias reduces and the variance increases. In other words, bias has a negative first-order derivative in response to model complexity while variance has a positive slope. The task of a Machine Learning practitioner is to find a good trade-off between the two.

Although in general Machine Learning applications the total number of errors matters, understanding bias and variance gives insight in the predictive behaviour of learning algorithms. Therefore, it is a very important aspect of meta-learning, potentially increasing both our knowledge about algorithms and the performance of newly created algorithms.

3.8 Discussion

This chapter reviewed aspects of meta-learning; techniques that improve knowledge of the learning process and techniques that help selecting an appropriate learning algorithm. The algorithm selection problem can be solved by learning techniques as well as search techniques. Often, the decision of which paradigm to chose comes down to whether experimentation with the data is allowed. When the data is not accessible yet or a recommendation must be provided relatively fast, the search paradigm is not applicable; hence the learning paradigm should be preferred. On the other hand, when there is time to try multiple configurations, the search paradigm is likely to recommend a good algorithm and parameter setting combination. Sometimes, the

search methods can be guided by some form of meta-learning.

The advantages of the learning approaches are plenty. First, the resulting meta-model is very general; it can be applied to all unseen problems. Second, algorithm selection for new tasks is rather inexpensive. Although building the meta-model can be a computational intensive task, applying it to new tasks happens typically fast. Finally, it is reasonable to expect the first attempted solution already to yield good result. Apart from calculating the meta-features, there are no start-up costs.

However, this approach also has some intrinsic limitations. First, it is hard to construct a meta-feature set that adequately characterizes the problem space [86]. Second, the most successful meta-features, landmarkers, can be computationally expensive, limiting the options [108]. Finally, because not all classifiers run on all datasets, or take prohibitively long to do so, the meta-dataset usually contains many missing values, complicating the classification task.

The search paradigm can be seen as a more thorough alternative. First, as it naturally tries multiple promising candidates, it will return a good solution with high probability. In the multi-armed bandit approach, there are even theoretical guarantees about the performance. Second, it is an iterative approach, that is constantly expected to improve itself. After a few iterations we can already expect reasonable recommendations, and it is likely that these keep on improving. Third, most search methods do not require a meta-dataset. As it is considered a laborious task to create an appropriate meta-dataset, it is convenient that search methods do not require such.

There are also limitations, compared to the learning paradigm. Most prominently, it comes with additional run time. Search methods are built upon function evaluations, i.e., multiple models are built and evaluated. Furthermore, obtained results do not generalize. The search procedure needs to be executed for each task again. Nothing is learned about the interaction between the mechanism of learning and the concrete contexts in which that mechanism is applicable. In that sense, this form of algorithm selection does not obey the definition of meta-learning given by Vilalta et al. [159].

There are some additional challenges that meta-learners are faced with. Many algorithm selection problems are subject to the curse of dimensionality [152]. Various studies proposed many different types of meta-features, yet the amount of available datasets is relatively small. Although many datasets exists, most of these are in people's labs and heads, not available to the community. Furthermore, there is a high computational cost for each instance. Indeed, all algorithms should be ran on it, which takes a lot of resources. For these reasons, experiment databases have been proposed [124, 153, 154, 155]. These aim to store and organise datasets and results from earlier experiments, available to the whole community. In the next chapter we

will review how Machine Learning and meta-learning research can benefit from such infrastructures.