1 Introduction

Summary. This chapter starts by describing the organization of the book, which consists of three parts. Part I discusses some basic concepts, including, for instance, what metalearning is and how it is related to automatic machine learning (AutoML). This continues with a presentation of the basic architecture of metalearning/AutoML systems, discussion of systems that exploit algorithm selection using prior metadata, methodology used in their evaluation, and different types of meta-level models, while mentioning the respective chapters where more details can be found. This part also includes discussion of methods used for hyperparameter optimization and workflow design. Part II includes the discussion of more advanced techniques and methods. The first chapter discusses the problem of setting up configuration spaces and conducting experiments. Subsequent chapters discuss different types of ensembles, metalearning in ensemble methods, algorithms used for data streams and transfer of meta-models across tasks. One chapter is dedicated to metalearning for deep neural networks. The last two chapters discuss the problem of automating various data science tasks and trying to design systems that are more complex. Part III is relatively short. It discusses repositories of metadata (including experimental results) and exemplifies what can be learned from this metadata by giving illustrative examples. The final chapter presents concluding remarks.

1.1 Organization of the Book

This book comprises three parts. In Part I (Chaps. 2–7) we sketch the basic concepts and architecture of metalearning systems, especially focusing on which types of "metaknowledge" can be collected by observing the performance of different models on prior tasks and how this can be used within metalearning approaches to learn new tasks more efficiently. Since this type of metalearning is closely related to automated machine learning (AutoML), we also cover this topic in some depth, but with a specific focus on how we can improve AutoML through metalearning.

Part II (Chaps. 8–15) covers different extensions of these basic ideas to more specific tasks. First, we discuss some methods that can be used in the design of configuration spaces that affect the search of metalearning and AutoML systems. Then, we show how metalearning can be used to build better ensembles and to recommend algorithms for streaming data. Next, we discuss how to transfer information from previously learned models to new tasks, using transfer learning and few-shot learning in neural networks.

P. Brazdil et al., *Metalearning*, Cognitive Technologies,
https://doi.org/10.1007/978-3-030-67024-5_1

The final two chapters are dedicated to the problem of automating data science and the design of complex systems.

Part III (Chaps. 16–18) provides practical advice on how to organize metadata in repositories and how this can be leveraged in machine learning research. The last chapter includes our closing remarks on and presents future challenges.

1.2 Basic Concepts and Architecture (Part I)

1.2.1 Basic Concepts

Role of machine learning

We are surrounded by data. On a daily basis, we are confronted by many forms of it. Companies try to market their products with commercials in the form of billboards and online advertisements. Large sensor networks and telescopes measure complex processes, happening around us on Earth or throughout the universe. Pharmaceutical institutions record the interactions between types of molecules, in search of new medicaments for new diseases.

All this data is valuable, as it enables us to characterize different situations, learn to separate them into different groups, and incorporate this into a system that can help us make decisions. We can thus identify fraudulent transactions from financial data, develop new medical drugs based on clinical data, or speculate about the evolution of celestial bodies in our universe. This process involves *learning*.

The scientific community has elaborated many techniques for analyzing and processing data. A traditional scientific task is modeling, where the aim is to describe the given complex phenomenon in a simplified way, in order to learn something from it. Many data modeling techniques have been developed for that purpose based on various intuitions and assumptions. This area of research is called *Machine Learning*.

Role of metalearning

As was shown, we cannot assume that there is one algorithm that works for all sorts of data, as each algorithm has its own area of expertise. Selecting the proper algorithm for a given task and dataset is key to obtaining an adequate model. This, in itself, can be seen as a learning task.

This process of learning across tasks has generally been called *metalearning*. Over the past decades, however, various machine learning researchers have used this term in many different ways, covering concepts such as meta-modeling, learning to learn, continuous learning, ensemble learning, and transfer learning. This large and growing body of work has clearly demonstrated that metalearning can make machine learning drastically more efficient, easier, and more trustworthy.

The area of metalearning is a very active research field. Many new and interesting research directions are emerging that address this general goal in novel ways. This book is meant to provide a snapshot of the most established research to date. As the area has grown a lot, the material had to be organized and structured into cohesive units. For instance, dataset characteristics are discussed in a separate chapter (Chapter 4), even though they play an important role in many other chapters.

Definition of metalearning

Let us start with a definition of metalearning, as it is viewed in this book:

> Metalearning is the study of principled methods that exploit metaknowledge to obtain efficient models and solutions by adapting machine learning processes.

The *metaknowledge* referred to above typically includes any sort of information obtained from previous tasks, such as descriptions of prior tasks, the pipelines and neural architectures tried, or the resulting models themselves. In many cases, it also includes knowledge that is obtained *during* the search for the best model on a new task, and that can be leveraged to guide the search for better learning models. Lemke et al. (2015) describe this from a systems point of view:

> A metalearning system must include a learning subsystem, which adapts with experience. Experience is gained by exploiting meta-knowledge extracted: a) in a previous learning episode on a single dataset and/or b) from different domains or problems.

Currently, the aim of many is to exploit the metadata gathered both on the past and the target dataset.

Metalearning versus automated machine learning (AutoML)

One important question is: what is the difference between a metalearning system and an AutoML system? Although this is a rather subjective matter for which different answers may be given, here we present the definition of AutoML of Guyon et al. (2015):

> AutoML refers to all aspects of automating the machine learning process, beyond model selection, hyperparameter optimization, and model search, . . .

Many AutoML systems make use of experience obtained from previously seen datasets. As such, many AutoML systems are, according to the definition above, also metalearning systems. In this book, we focus on techniques that involve metalearning, as well as AutoML systems that often use metalearning.

On the origins of the term *metalearning*

The pioneering work of Rice (1976) is discussed in Chapter 1. This work was not widely known in the machine learning community until much later. In the 1980s Larry Rendell published various articles on *bias management* (this topic is discussed in Chapter 8). One of his articles (Rendell et al., 1987) includes the following text:

> The VBMS [Variable Bias Management System] can perform **meta-learning**. Unlike most other learning systems, the VBMS learns at different levels. In the process of learning a concept the system will also acquire knowledge about induction problems, biases, and the relationship between the two. Thus the system will not only learn concepts, but will also learn about the relationship between problems and problem-solving techniques.

P. Brazdil encountered the term *meta-interpreter* in connection with the work of Kowalski (1979) at the University of Edinburgh in the late 70’s. In 1988 he organized a workshop on *Machine Learning, Meta-Reasoning and Logics* (Brazdil and Konolige, 1990). The introduction of this book includes the following passage:

> For some meta-knowledge represents knowledge that talks about other (object level) knowledge. The purpose of meta-knowledge is mainly to control inference. In the second school of thought, **meta-knowledge** has a somewhat different role: it is used to control the process of knowledge acquisition and knowledge reformulation (learning).

Metalearning was explored in project StatLog (1990–93) (Michie et al., 1994).

1.2.2 Problem types

The scientific literature typically distinguishes the following problem types, many of which will be referred to throughout the book. The general aim of metalearning systems is to learn from the usage of prior models (how they were constructed and how well they performed) in order to *to model a target dataset better*. If the base-level task is classification, this implies that the system can predict the value of the target variable, i.e. the class value in this case. Ideally, it does this better (or more efficiently) by leveraging information besides the training data itself.

Algorithm selection (AS): Given a set of algorithms and a dataset (target dataset), determine which algorithm is most appropriate to model the target dataset.

Hyperparameter optimization (HPO): Given an algorithm with specific *hyperparameters* and a target dataset, determine the best hyperparameter settings of the given algorithm to model the target dataset.

Combined algorithm selection and hyperparameter optimization (CASH): Given a set of algorithms, each with its own set of hyperparameters, and a target dataset, determine which algorithm to use and how to set its hyperparameters to model the target dataset. Some CASH systems address also the more complex pipeline synthesis task discussed next.

Workflow (pipeline) synthesis: Given a set of algorithms, each with its own set of hyperparameters, and a target dataset, design a workflow (pipeline) consisting of a one or more algorithms to model the target dataset. The inclusion of a particular algorithm and its hyperparameter settings in the workflow can be seen as a CASH problem.

Architecture search and/or synthesis: This problem type can be seen as a generalization of the problem type above. In this setting the individual constituents do not need to be organized in a sequence, as it is done in workflows (pipelines). The architecture can include, for instance, partially ordered or tree-like structures. The neural network architecture design can be seen as a problem that falls into this category.

Few-shot learning: Given a target dataset with few examples and various datasets that are very similar, but include many examples, retrieve a model that has been pretrained on prior datasets and fine-tune it to perform well on the target dataset.

We note that algorithm selection problems are defined on a *discrete* set of algorithms, while the hyperparameter optimization problem and CASH problem are typically defined on *continuous* configuration spaces, or heterogeneous spaces with both discrete

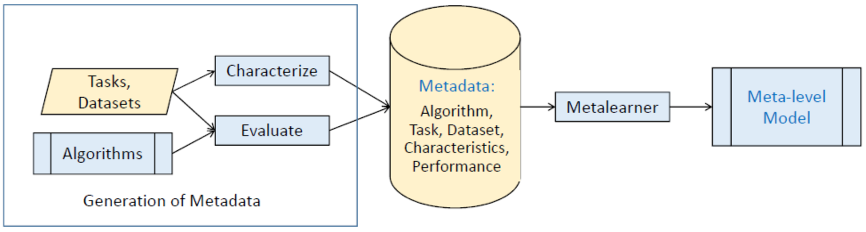

Fig. 1.1: Generating a meta-level model

and continuous variables. Techniques for algorithm selection can also easily be applied to discretized versions of the latter.

In this book, we follow the following convention that has been broadly adopted by the machine learning community. A *hyperparameter* is a user-defined parameter that determines the behavior of a particular machine learning algorithm; for instance, the level of pruning in a decision tree, or the learning rate in neural networks are hyperparameters. A (model) *parameter* is a parameter that has been learned based on the training data. For instance, the weights of a neural network model are regarded as model parameters.

1.2.3 Basic architecture of metalearning and AutoML systems

The algorithm selection problem was first formulated by Rice (1976). He has observed that it is possible to relate the performance of algorithms to *dataset characteristics/features*. In other words, dataset features are often quite good predictors of performance of algorithms. This can be exploited to identify the best performing algorithm for a given target dataset. It has since then be applied to many application domains, also beyond machine learning (Smith-Miles, 2008).

A general architecture for metalearning systems that address the algorithm selection problem is shown in Figure 1.1. First, *metadata* is collected that encodes information on previous learning episodes. This includes descriptions of the *tasks* that we solved before. For instance, these could be classification tasks where we build classifiers for a given dataset, or reinforcement learning tasks defined by different learning environments. *Characterizations* of these tasks are often very useful for reasoning how new tasks could be related to prior tasks. The metadata also includes *algorithms* (e.g. machine learning pipelines or neural architectures) that were previously used for learning these tasks, and information about performance resulting from *evaluations*, showing how well that worked. In some cases, we can store the trained models as well, or measurable properties of these models. Such metadata from many previous (or current) tasks could be combined in a database, or a “memory” of prior experience that we want to build on.

We can leverage this metadata in many different ways. For instance, we could use metadata directly inside metalearning algorithms, or use it to train a meta-level model. Such a meta-model can be used inside a metalearning system, as illustrated in Figure 1.2. Based on the characteristics of a novel “target” task, the meta-model could construct or recommend new algorithms to try, and use the observed performance to update the algorithms or recommendations until some stopping condition, usually a time budget or performance constraint, has been satisfied. In some cases, no task characteristics are included, and the metalearning system learns from prior experience and observations on the new task alone.

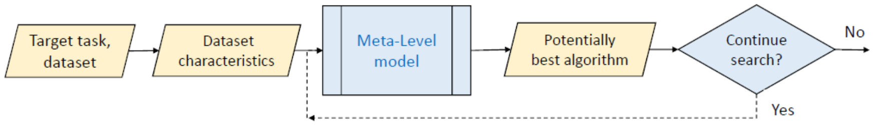

Fig. 1.2: Using a meta-level model to predict the best algorithm

1.2.4 Algorithm selection using metadata from prior datasets (Chaps. 2,5)

Chapters 2 and 5 discuss methods that exploit performance metadata of algorithms on previous tasks to recommend algorithms for a target dataset. These recommendations can take the form of rankings of candidate algorithms (Chap. 2) or meta-models that predict the suitability of algorithms on new tasks (Chap. 5).

In Chapter 2 we describe a relatively simple approach, the *average ranking method*, which is often used as a baseline method. The average ranking algorithm uses meta-knowledge acquired in previous tasks to identify the potentially best base-level algorithms for the current task.

This approach requires that an appropriate evaluation measure, such as *accuracy*, is set beforehand. In this chapter we also describe a method that builds this ranking based on a combination of accuracy and runtime, yielding good anytime performance.

Chapter 5 discusses more advanced methods. The methods in both chapters are designed to work on discrete problems.

1.2.5 Evaluation and comparisons of different systems (Chap. 3)

When working with a given metalearning system, it is important to know whether we can trust its recommendations and how its performance compares with other competing approaches. Chapter 3 discusses a typical approach that is commonly used to evaluate metalearning systems and make comparisons.

In order to obtain a good estimate of the performance of a metalearning system, it needs to be evaluated on many datasets. As the performance of algorithms may vary substantially across these datasets, many systems normalize the performance values first to make comparisons meaningful. Section 3.1 discusses some of the most common normalization methods used in practice.

Assuming that a metalearning system outputs a sequence of algorithms to test, we can study how similar this sequence is from the ideal sequence. This can be determined by looking at a degree of correlation between the two sequences. Section 3.2 describes this in more detail.

We note that the above approach compares the predictions made by meta-level model with the meta-target (i.e., correct ordering of algorithms). A disadvantage of this approach is that it does not show directly the effects in terms of base-level performance. This problem can be avoided by considering the appropriate base-level performance of different metalearning systems and how this evolves with time. If the ideal performance is known, it is possible to calculate the value of *performance loss*, which is the difference between the actual performance and the ideal value. The loss curve shows how the loss evolves with time. In some systems the maximum available time (i.e., time budget) is given beforehand. Different systems and their variants can then be compared by looking at how the loss evolves with time. More details can be found in Section 3.3.

Section 3.4 also presents some useful measures, such as *loose accuracy* and *discounted cumulative gain* that are often used when comparing sequences. The final section (Section 3.5) describes the methodology that is commonly used in comparisons involving several metalearning/AutoML systems.

1.2.6 Role of dataset characteristics/metafeatures (Chap. 4)

We note that in the proposal of Rice (1976), dataset characteristics play a crucial role. They have been used in many metalearning systems since then. Typically, they help to restrict the search for the potentially best algorithm. If the characteristics are not available, or if they are difficult to define in a given domain, the search can proceed nevertheless. The basic approaches based on rankings or pairwise comparisons discussed in Chapters 2 and 5 can be used without any dataset characteristics. This is an advantage, as indeed, in many domains, it is difficult to come up with a sufficient number of informative characteristics that enable to discriminate a large number of very similar algorithm configurations.

One important group of characteristics is the group of *performance-based characteristics*. This group includes, for instance, *sampling landmarkers*, representing the performance of particular algorithms on samples of data. These can be obtained in virtually all domains.

There is no doubt that some characteristics are, in general, important. Consider, for instance, the basic characteristic of the target variable. If it is numeric, it suggests that a suitable regression algorithm should be used, while if it is categorical, a classification algorithm should be used. A similar argument can be given when we encounter balanced/unbalanced data. This characteristic conditions the choice of the right approach.

Chapter 4 discusses various dataset characteristics, organized by task type, such as classification, regression, or time series. Various chapters in this book discuss how the datasets characteristics are effectively used in different metalearning approaches (e.g., Chaps. 2, 5).

1.2.7 Different types of meta-level models (Chap. 5)

Several types of meta-level models were used in the past:

- regression model,
- classification model,
- relative performance model.

Chapter 5 discusses the details of such models. These are used in various approaches discussed throughout the book.

The regression model uses a suitable regression algorithm which is trained on the metadata and can then be used to predict the performance of a given set of base-level algorithms. The predictions can be used to order the base-level algorithms and hence identify the best one.

Regression models play an important role also in the search for the best hyperparameter configuration, particularly if these are numeric and continuous. For example, the method called *sequential model-based optimization* discussed in Chapter 6 uses a regression algorithm on a meta-level to model the loss function, and to identify promising hyperparameter settings.

A classification model identifies which of the base-level algorithms are *applicable* to the target classification task. This implies that these algorithms are likely to obtain a relatively good performance on the target task. We note that this metalearning task is applied to a discrete domain.

If we were to use probabilistic classifiers at the meta-level which provide apart from the class (e.g., applicable or non-applicable), also numeric values related to the probability of classification, then these values can be used to identify the potentially best possible base-level algorithm or to explore a ranking in further search.

The relative performance model is based on an assumption that it is not necessary to have the details about the actual performance of algorithms, if the aim is to identify good-performing algorithms. All that is needed is the information regarding their relative performance. Relative performance models can either use rankings or pairwise comparisons. In all these settings it is possible to use search to identify the potentially best algorithm for the target dataset.

1.2.8 Hyperparameter optimization (Chap. 6)

Chapter 6 describes various approaches for the hyperparameter optimization and combined algorithm selection and hyperparameter optimization problems.

This chapter differs from the Chapters 2 and 5 in one important aspect: it discusses methods that use performance metadata obtained mainly on the target dataset. The metadata is used to construct relatively simple and fast-to-test models of the target algorithm configuration (algorithm with appropriate hyperparameter settings) that can be queried. The aim of these queries is to identify the best configuration to test, which is the one with the highest estimate of performance (e.g., accuracy). This type of search is referred to as *model-based search*.

The situation is not entirely clear-cut, though. As is shown, the metadata gathered on past datasets can also be useful and improve the performance of model-based search.

1.2.9 Automatic methods for workflow design (Chap. 7)

Many tasks require a solution which does not involve a single base-level algorithm, but rather several algorithms. The term *workflow* (or *pipeline*) is often used to represent such sequences. In general, the set may be only partially ordered.

When designing workflows (pipelines), the number of configurations can grow dramatically. This is because each item in the workflow can in principle be substituted by an appropriate base-level operation and there may be several such operations available. The problem is exacerbated by the fact that a sequence of two or more operators can in general be executed in any order, unless instructions are given to the contrary. This creates a problem, as for N operators there are $N!$ possible orderings. So, if a set of operators should be executed in a specific order, explicit instructions need to be given to that effect. If the order is irrelevant, the system should also be prevented from experimenting with alternative orderings. All alternative workflows and their configurations (including all possible hyperparameter settings) constitute the so-called *configuration space*.

Chapter 7 discusses various means that have been used to restrict the design options and thus reduce the size of the configuration space. These include, for instance, ontologies and context-free grammars. Each of these formalisms has its merits and shortcomings.

Many platforms have resorted to planning systems that use a set of operators. These can be designed to be in accordance with given ontologies or grammars. This topic is discussed in Section 7.3.

As the search space may be rather large, it is important to leverage prior experience. This topic is addressed in Section 7.4. which discusses *rankings of plans* that have proved to be useful in the past. So, the workflows/pipelines that have proved successful in the past, can be retrieved and used as plans for future tasks. So, it is possible to exploit both planning and metalearning.

1.3 Advanced Techniques and Methods (Part II)

1.3.1 Setting up configuration spaces and experiments (Chap. 8)

One of the challenges that metalearning and AutoML research faces nowadays is that the number of algorithms (workflows in general) and their configurations is so large that it can cause problems when searching through this space for an acceptable solution. Also, it is not possible any more to have a complete set of experimental results (complete metadata). So, several questions arise:

1. Is the configuration space adequate for the set of tasks of interest? This question is addressed in Section 8.3.
2. Which parts of the configuration space are relevant, and which parts are less relevant? This question is addressed in Section 8.4.
3. Can we reduce the configuration space to make metalearning more effective? This question is addressed in Section 8.5.

Considering this from the perspective of the algorithm selection framework, these questions are concerned with the algorithm space.

In order to successfully learn, also several aspects from the problem space become important. We address the following questions:

1. Which datasets do we need to be able to transfer knowledge to new datasets? This question is addressed in Section 8.7.
2. Do we need complete metadata, or does incomplete metadata suffice? This question is already partly addressed in Chapter 2, and is further elaborated on in Section 8.8.
3. Which experiments need to be scheduled first to obtain adequate metadata? This question is addressed in Section 8.9.

1.3.2 Automatic methods for ensembles and streams

Combining base-learners into ensembles (Chap. 9)

Ensembles of classification or regression models represent an important area of machine learning. They have become popular because they tend to achieve high performance when compared with single models. This is why we devote one chapter to this topic in this book. We start by introducing ensemble learning and present an overview of some of its most well-known methods, including bagging, boosting, stacking, and cascade generalization, among others.

Metalearning in ensemble methods (Chap. 10)

There is a growing number of approaches integrating metalearning methods – in the sense used in this book – in ensemble learning approaches.[1] In Chapter 10 we discuss some of those approaches. We start by providing a general perspective and then we analyze them in detail, concerning the ensemble method used, the metalearning approach employed, and finally the metadata involved.

We show that ensemble learning presents many opportunities for research on metalearning, with very interesting challenges, namely in terms of the size of the configuration space, the definition of the competence regions of the models, and the dependency between them. Given the complexity of ensemble learning systems, one final challenge is to apply metalearning to understand and explain their behavior.

Algorithm recommendation for data streams (Chap. 11)

Real-time analysis of data streams is a key area of data mining research. Many real-world collected data is in fact a stream where observations come in one by one, and algorithms processing these are often subject to time and memory constraints. Examples of this are stock prices, human body measurements, and other sensor measurements. The nature of data can change over time, effectively outdating models that we have built in the past.

This has been identified by the scientific community, and hence many machine learning algorithms have been adapted or are specifically designed to work on data streams. Some examples of this are *Hoeffding trees*, *online boosting*, and *leveraging bagging*. Also, the scientific community provided the so-called *drift detectors*, mechanisms that identify when the created model is no longer applicable. Once again, we are faced with an algorithm selection problem, which can be solved with metalearning.

In this chapter, we discuss three approaches on how techniques from within the scope of this book have been used to address this problem. First, we discuss metalearning approaches that divide the streams into various intervals, calculate metafeatures over these parts, and use a meta-model for each part of the stream to select which classifier to use. Second, we discuss ensemble approaches, that use the performance on recent data to determine which ensemble members are still up to date. In some sense, these methods are much simpler to apply in practice, as they do not rely on a basis of metalearning, and consistently outperform the metalearning approaches. Third, we discuss approaches

[1]In ensemble learning literature the term *metalearning* is used to refer to certain ensemble learning approaches (Chan and Stolfo, 1993), where it has a somewhat different meaning from the one used in this book.

that are built upon the notion of recurring concepts. Indeed, it is reasonable to assume some sort of seasonality in data, and models that have become outdated can become relevant again at some point later in time. This section describes systems that facilitate this type of data. Finally, this chapter closes with open research questions and directions for future work.

1.3.3 Transfer of meta-models across tasks (Chap. 12)

Many people hold the view that learning should not be viewed as an isolated task that starts from scratch with every new problem. Instead, a learning algorithm should exhibit the ability to exploit the results of previous learning processes to new tasks. This area is often referred to as *transfer of knowledge across tasks*, or simply *transfer learning*. The term *learning to learn* is also sometimes used in this context.

Chapter 12 is dedicated to transfer learning, whose aim is to improve learning by detecting, extracting, and exploiting certain information across tasks. This chapter is an *invited chapter* written by Ricardo Vilalta and Mikhail M. Meskhi, with the aim of complementing the material of this book.

The authors discuss different approaches available to transfer knowledge across tasks, namely representational transfer and functional transfer. The term *representational transfer* is used to denote cases when the target and source models are trained at different times and the transfer takes place after one or more source models have already been trained. In this case there is an explicit form of knowledge transferred directly to the target model or to a meta-model that captures the relevant part of the knowledge obtained in past learning episodes.

The term *functional transfer* is used to denote cases where two or more models are trained simultaneously. This situation is sometimes referred to as *multi-task learning*. In this case the models share (possibly a part of) their internal structure during learning. More details on this can be found in Section 12.2.

The authors address the question regarding what exactly can be transferred across tasks and distinguish instance-, feature-, and parameter-based transfer learning (see Section 12.3). Parameter-based transfer learning describes a case when the parameters found on the source domain can be used to initialize the search on the target domain. We note that this kind of strategy is also discussed in Chapter 6 (Section 6.7).

As neural networks play an important role in AI, one whole section (Section 12.3) is dedicated to the issue of transfer in neural networks. So, for instance, one of the approaches involves transfer of a part of network structure. This section also describes a *double loop architecture*, where the base-learner iterates over the training set in an inner loop, while the metalearner iterates over different tasks to learn metaparameters in an outer loop. This section also describes transfer in kernel methods and in parametric Bayesian models. The final section (Section 12.4) describes a theoretical framework.

1.3.4 Metalearning for deep neural networks (Chap. 13)

Deep learning methods are attracting a lot of attention recently, because of their successes in various application domains, such as image or speech recognition. As training is generally slow and requires a lot of data, metalearning can offer a solution to this problem. Metalearning can help to identify the best settings for hyperparameters, as well as parameters, concerned, for instance, with weights of a neural network model.

Most metalearning techniques involve a learning process at two levels, as was already pointed out in the previous chapter. At the *inner level*, the system is presented with a new task and tries to learn the concepts associated with this task. This adaptation is facilitated by the knowledge that the agent has accumulated from other tasks, at the *outer level*.

The authors categorize this field of metalearning in three groups: metric-, model-, and optimization-based techniques, following previous work. After introducing notation and providing background information, this chapter describes key techniques of each category, and identifies the main challenges and open questions. An extended version of this overview is also available outside this book (Huisman et al., 2021).

1.3.5 Automating data science and design of complex systems

Automating data science (AutoDS) (Chap. 14)

It has been observed that in data science a greater part of the effort usually goes into various preparatory steps that precede model-building. The actual model-building step typically requires less effort. This has motivated researchers to examine how to automate the preparatory steps and gave rise to methodology that is known under the name *CRISP-DM model* (Shearer, 2000).

The main steps of this methodology include problem understanding and definition of the current problem/task, obtaining the data, data preprocessing and various transformations of data, model building, and its evaluation and automating report generation. Some of these steps can be encapsulated into a workflow, and so the aim is to design a workflow with the best potential performance.

The model-building step, including hyperparameter optimization, is discussed in Chapter 6. More complex models in the form of workflows are discussed in Chapter 7. The aim of Chapter 14 is to focus on the other steps that are not covered in these chapters.

The area related to the definition of the current problem (task) involves various steps. In Section 14.1 we argue that the problem understanding of a domain expert needs to be transformed into a description that can be processed by the system. The subsequent steps can be carried out with the help of automated methods. These steps include generation of task descriptors (e.g., keywords) that help to determine the task type, the domain, and the goals. This in turn allows us to search for and retrieve domain-specific knowledge appropriate for the task at hand. This issue is discussed in Section 14.2.

The operation of obtaining the data and its automation may not be a trivial matter, as it is necessary to determine whether the data already exists or not. In the latter case a plan needs to be elaborated regarding how to get it. Sometimes it is necessary to merge data from different sources (databases, OLAP cube, etc.). Section 14.3 discusses these issues in more detail.

The area of preprocessing and transformation has been explored more by various researchers in the AutoDS community. Methods exist for selection of instances and/or elimination of outliers, discretization and various other kinds of transformations. This area is sometimes referred to as *data wrangling*. These transformations can be learned by exploiting existing machine learning techniques (e.g., learning by demonstration). More details can be found in Section 14.3.

One other important are of data science involves decisions regarding the appropriate level of detail to be used in the application. As has been shown, data may be summarized by appropriate aggregation operations, such as *Drill Down/Up* operations in a given OLAP cube. Categorical data may also be transformed by introducing new high-level features. This process involves determining the right level of granularity. Efforts can be made to automate this, but more work is needed before it is possible to offer practical solutions to companies. More details about this issue can be found in Section 14.4.

Automating the design of complex systems (Chap. 15)

In this book we have dealt with the problem of automating the design of KDD workflows and other data science tasks. A question arises regarding whether the methods can be extended to somewhat more complex tasks. Chapter 15 discusses these issues, but the focus in this book is on symbolic approaches.

We are well aware that many successful applications nowadays, particularly in vision and NLP, exploit deep neural networks (DNNs), CNNs, and RNNs. Despite this, we believe that symbolic approaches continue to be relevant. We believe that this is the case for the following reasons:

- DNNs typically require large training data to work well. In some domains not many examples are available (e.g., occurrences of rare diseases). Also, whenever the examples are supplied by a human (as, for instance, in data wrangling discussed in Chapter 14), we wish the system to capable of inducing the right transformation on the basis of a modest number of examples. Many systems in this area exploit symbolic representations (e.g., rules), as it is easy to incorporate background knowledge, which is often also in the form of rules.
- It seems that, whenever AI systems need to communicate with humans, it is advantageous to resort to symbolic concepts which can easily be transferred between a human and a system.
- As human reasoning includes both symbolic and subsymbolic parts, it is foreseeable that future AI systems will follow this line too. So it is foreseeable that the two reasoning systems will coexist in a kind of functional symbiosis. Indeed, one current trend involves so-called *explainable AI*.

The structure of this chapter is as follows. Section 15.1 discusses more complex operators that may be required when searching for a solution of more complex tasks. This includes, for instance, conditional operators and operators for iterative processing.

Introduction of new concepts is addressed in Section 15.2. It discusses changes of granularity by the introduction of new concepts. It reviews various approaches explored in the past, such as, constructive induction, propositionalization, reformulation of rules, among others. This section draws attention to some new advances, such as feature construction in deep NNs.

There are tasks that cannot be learned in one go, but rather require a sub-division into subtasks, a plan for learning the constituents, and joining the parts together. This methodology is discussed in Section 15.3. Some tasks require an iterative process in the process of learning. More details on this can be found in Section 15.4. There are problems whose tasks are interdependent. One such problem is analyzed in Section 15.5.

1.4 Repositories of Experimental Results (Part III)

1.4.1 Repositories of metadata (Chap. 16)

Throughout this book, we discuss the benefits of using knowledge about past datasets, classifiers, and experiments. All around the globe, thousands of machine learning experiments are being executed on a daily basis, generating a constant stream of empirical information on machine learning techniques. Having the details of experiments freely available to others is important, as it enables to reproduce the experiments and verify that the conclusions are correct and use this knowledge to extend the work further. It can thus speed up progress in science.

This chapter starts with a review of online repositories where researchers can share data, code, and experiments. In particular, it covers OpenML, an online platform for sharing and organizing machine learning data automatically and in fine detail. OpenML contains thousands of datasets and algorithms, and millions of experimental results on these experiments. In this chapter we describe the basic philosophy behind it, and its basic components: datasets, tasks, flows, setups, runs, and benchmark suites. OpenML has API bindings in various programming languages, making it easy for users to interact with the API in their native language. One hallmark feature of OpenML is the integration into various machine learning toolboxes, such as Scikit-learn, Weka, and mlR. Users of these toolboxes can automatically upload all the results that they obtained, leading to a large repository of experimental results.

1.4.2 Learning from metadata in repositories (Chap. 17)

Having a vast set of experiments, collected and organized in a structured way, allows us to conduct various types of experimental studies. Loosely based upon a categorization of Vanschoren et al. (2012), we present three types of experiments that explore OpenML metadata to investigate certain issues to be described shortly: experiments on a single dataset, on multiple datasets, and experiments involving specific dataset or algorithm characterization.

As for the experiments on a single dataset, Section 17.1 shows how the OpenML metadata can be used for simple benchmarking and, in particular, to assess the impact of varying the settings of a specific hyperparameter. As for the experiments on multiple datasets, Section 17.2 shows how the OpenML metadata can be used to assess the benefit of hyperparameter optimization, and also, the differences on predictions between algorithms. Finally, for the experiments involving specific characteristics, Section 17.3 shows how the OpenML metadata can be used to investigate and answer certain scientific hypotheses, such as on what type of datasets are linear models adequate, and for what type of datasets feature selection is useful. Furthermore, we present studies whose aim is to establish the relative importance of hyperparameters across datasets.

1.4.3 Concluding remarks (Chap. 18)

The last chapter of the book (Chap. 18) presents the concluding remarks with respect to the whole book. It includes two sections. As metaknowledge has a central role in many approaches discussed in this book, we analyze this issue in more detail here. In particular, we are concerned with the issue of what kind of metaknowledge is used in different metalearning/AutoML tasks, such as algorithm selection, hypeparameter optimization, and workflow generation. We draw attention to the fact that some metaknowledge is acquired (learned) by the systems, while other is given (e.g., different aspects of the given configuration space). More details on this can be found in Section 18.1.

Section 18.2 discusses future challenges, such as better integration of metalearning and AutoML approaches and what kind of guidance could be provided by the system for the task of configuring metalearning/AutoML systems to new settings. This task involves (semi-)automatic reduction of configuration spaces to make the search more effective. The last part of this chapter discusses various challenges which we encounter when trying to automate different steps of data science.

References

Brazdil, P. and Konolige, K. (1990). *Machine Learning, Meta-Reasoning and Logics*. Kluwer Academic Publishers.

Chan, P. and Stolfo, S. (1993). Toward parallel and distributed learning by meta-learning. In *Working Notes of the AAAI-93 Workshop on Knowledge Discovery in Databases*, pages 227–240.

Guyon, I., Bennett, K., Cawley, G., Escalante, H. J., Escalera, S., Ho, T. K., Macià, N., Ray, B., Saeed, M., Statnikov, A., et al. (2015). Design of the 2015 ChaLearn AutoML challenge. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Huisman, M., van Rijn, J. N., and Plaat, A. (2021). A survey of deep meta-learning. *Artificial Intelligence Review*.

Kowalski, R. (1979). *Logic for Problem Solving*. North-Holland.

Lemke, C., Budka, M., and Gabrys, B. (2015). Metalearning: a survey of trends and technologies. *Artificial Intelligence Review*, 44(1):117–130.

Michie, D., Spiegelhalter, D. J., and Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

Rendell, L., Seshu, R., and Tcheng, D. (1987). More robust concept learning using dynamically-variable bias. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 66–78. Morgan Kaufmann Publishers, Inc.

Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, 15:65–118.

Shearer, C. (2000). The CRISP-DM model: the new blueprint for data mining. *J Data Warehousing*, 5:13–22.

Smith-Miles, K. A. (2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41(1):6:1–6:25.

Vanschoren, J., Blockeel, H., Pfahringer, B., and Holmes, G. (2012). Experiment databases: a new way to share, organize and learn from experiments. *Machine Learning*, 87(2):127–158.