18 Concluding Remarks

Summary. As metaknowledge has a central role in many approaches discussed in this book, we address 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). This chapter continues by discussing future challenges, such as how to achieve better integration of metalearning and AutoML approaches, and what kind of guidance could be provided by the system when configuring metalearning/AutoML systems to new settings. This task may involve (semi-)automatic reduction of configuration spaces to make the search more effective. The last part of this chapter discusses various challenges encountered when trying to automate different steps of data science.

18.1 Introduction

This chapter includes two sections. In the first one, we analyze various forms of metaknowledge used in different approaches discussed in this book. Our aim is to present a unified view on this topic. The second section presents some future challenges in the area of metalearning, with applications to automated machine learning (AutoML). Our aim is to help researchers find interesting and promising lines of work.

18.2 Form of Metaknowledge Used in Different Approaches

Different metalearning approaches discussed in this book can be seen as involving two levels, a base-level and a meta-level. The former can be seen as a repository of base-level solutions or partially constructed solutions to address real-world problems, such as diagnosing a patient or predicting the lifetime value of a customer. These would typically include some machine learning algorithm with configured hyperparameters. More complex solutions can be represented as workflows or pipelines of operations and may involve rather complex structures, such as ensembles or deep neural networks.

The meta-level involves different types of information useful in the process of identifying the best possible configured base-level algorithm/workflow/structure for the current task. Various approaches were described in this book regarding how this can be

P. Brazdil et al., *Metalearning*, Cognitive Technologies,
https://doi.org/10.1007/978-3-030-67024-5_18

done. Metaknowledge, i.e., knowledge about the behavior of base-level processes, plays an important role in this process. It is normally derived (or learned) by the metalearning system from the given metadata. Consequently, we will refer in this chapter to this metaknowledge as *learned (or acquired) metaknowledge*.

However, the functioning of metalearning systems is also affected by the application setup, which involves different entities, including:

- A portfolio of base-level algorithms
- Hyperparameters of base-level algorithms and their possible settings
- Ontologies/grammars/operators determining admissible combinations of the above elements
- Application-specific constraints, such as computational resources available, delivery time, and model explainability requirements

The description of these entities is, in this chapter, referred to as *configuration metaknowledge*. These concepts are important, as they determine the space of alternatives that the system can consider in its search for the potentially best one. Normally this metaknowledge is provided by the data scientist. However, as we have shown in Chapter 8, some parts of configuration metaknowledge can be refined by the system.

Our aim here is to revisit some of the approaches discussed in this book, namely:

- Algorithm selection approaches
- Hyperparameter configuration
- Workflow generation
- Knowledge transfer across deep neural networks

and analyze the form of metaknowledge used in each case, with a particular focus on both learned metaknowledge and configuration metaknowledge.

18.2.1 Metaknowledge in algorithm selection approaches

In the following subsections, we distinguish between approaches that use either *prior metadata*, i.e., metadata acquired solely on prior tasks, and *dynamic metadata*, both acquired on prior tasks and updated on the current task.

Ranking approaches that use prior metadata

Chapter 2 discusses ranking-based approaches to algorithm selection. These approaches use performance metadata obtained on different prior tasks/ datasets to construct an *average ranking*. This ranking is followed on the target dataset until a given time budget has been exhausted. Dataset characteristics can be used to preselect a subset of datasets most similar to the target dataset and, thus, guide the search using the relevant metadata. The best algorithm identified is used to obtain the predictions on the target dataset. So in this approach, the average ranking represents the learned metaknowledge, ready to be applied to the new task. This approach requires that appropriate configuration metaknowledge is given.

Despite its simplicity, this approach can suggest quite good algorithms or workflows for the target dataset and achieve good performance. This is due to the fact that the given configuration metaknowledge can include rather complex structures, such as ensembles or deep neural networks and various variants of these tuned on specific tasks. So if the new task is similar to one of the past tasks, a ready-made solution is available. One disadvantage of this method is that the configuration metaknowledge is in an extensional form (e.g., a ranking of workflows (pipelines)), and also, that the ranking is static.

Approaches that exploit dynamic metadata

Some approaches discussed in Chapter 5 were conceived to overcome one of the shortcomings associated with the previous approach. They exploit pairwise tests and performance-based dataset characteristics (metafeatures). So the performance metadata is updated as the tests proceed on the current dataset. In other words, the metadata is changing dynamically.

The approach of *active testing* uses *estimates of performance gain* to characterize pairs of models (trained algorithms). These estimates represent the expected difference between the performance of the two models. So the metaknowledge is in the form of such estimates. This information is used by the system to suggest algorithms or workflows that may obtain higher performance than the incumbent (current best candidate).

A combination of prior and dynamic metadata is also often used in the application of metalearning for ensemble methods, which is discussed in Chapter 10.

18.2.2 Metaknowledge in approaches for hyperparameter optimization

Chapter 6 discusses how metalearning can be used for hyperparameter optimization. Various approaches discussed there explore the fact that the "performance surface" of different algorithm configurations (configurations with different hyperparameter settings) is rather "smooth" for many algorithms.[1] Consequently, it is possible to construct a meta-level model (often referred to as a *surrogate model*) that can be used in the search for configurations which are likely to lead to higher performance. The use of surrogate models has the advantage that it enables to identify the promising new configurations fast.

The meta-level model could be seen as a part of the learned metaknowledge. This approach is dynamic, because as more configurations have been examined, the meta-level model becomes more refined. Thus the refined model is, in turn, capable of providing more reliable suggestions regarding which hyperparameter settings could lead to better performance. The final model identified represents also learned metaknowledge, albeit generated on a specific task.

Some approaches use metaknowledge learned both on prior datasets and on the current dataset in this process, as was shown in Chapter 6. Some surrogate models can be employed across tasks. They learn a similarity function over tasks and can therefore transfer knowledge about parts of the configuration that work well to similar tasks.

18.2.3 Metaknowledge in workflow design

As the space of alternative workflows can be potentially very large, it is difficult to rely on extensional approaches consisting of the enumeration of alternatives. Various means have been designed to overcome this problem. The representation adopted is *intensional*, permitting to generate various possible workflows or their configurations. Chapter 7 discusses some representations commonly used in this area. These include:

- Ontologies (see Section 7.2)
- Context free grammars (CFGs) (see Section 7.2)
- Abstract/concrete operators of a planning system (see Section 7.3)

[1]Some algorithms (e.g., SVM with a specific kernel) do not really have a smooth performance surface.

As shown in Chapter 7, each form has certain advantages and disadvantages. These representations normally embody the knowledge of data scientists, hence are a part of configuration metaknowledge.

As we have pointed out in Chapter 7, the process of searching for the potentially best workflow can be seen as a process of gradually refining the metaknowledge acquired on the current task (learned metaknowledge).

18.2.4 Metaknowledge in transfer learning and in deep neural networks

Transfer learning, discussed in Chapter 12, is concerned with transferring (parts of) models, such as, parameters (i.e., weights), from one task to another.

Both the area of transfer learning and deep neural networks discuss the use of pretrained models on a set of relatively homogeneous datasets (meta-examples). This model can then be used (more precisely, its parameters) as the initial model on the target dataset (meta-example) (see, e.g., MAML discussed in Chapter 13). This model is then fine-tuned with new instances from the new task. This leads to satisfactory performance, even in problems where the number of available (labeled) instances is small. This scenario is often called *few-shot learning*.

So, if we use the terms used in the chapter, we note that the initial model (or the parameter weights) represent learned metaknowledge that is transferred to the target task. Although this still requires some training procedure on data from the target task, the required amount of data and time are often eminently smaller than when one would start without metaknowledge.

18.3 Future Challenges

Recent years have witnessed a large growth in research in the areas of metalearning and AutoML as well as their integration into data science tools. However, many interesting and relevant problems remain unsolved, or better solutions could be developed. In the following subsections we mention some of them.

18.3.1 Design of metafeatures relating dataset characteristics and performance

Preselecting datasets according to their similarity to the target dataset involves one of the most complex challenges in the development of metalearning systems: designing metafeatures that represent characteristics of the datasets that affect the (relative) performance of algorithms, as discussed in Chapter 4. Despite the different types of metafeatures that have been proposed and some work on the systematization of their design and usage, new useful contributions could still be made.

18.3.2 Further integration of metalearning and AutoML approaches

In Section 6 we have discussed some approaches whose aim was to explore both metaknowledge acquired on past problems and metaknowledge acquired on the current problem. The former capture general information about the behavior of learning processes, while the latter enables an adjustment of the solution to the current task. It is foreseeable that these approaches could be further improved. So the question is: what is the best method of integrating the two types of metaknowledge? This question can be addressed along different lines. Some are discussed in the following sections.

18.3.3 Automating the adaptation to the current task

It seems desirable that metalearning/AutoML systems should have the ability to carry out automatic adaptation (self-configuration) to the current task. This could involve, for instance, identifying the appropriate domain-specific knowledge, or learned metaknowledge that was acquired on similar tasks. Additionally, metaknowledge from the current task could enrich the metaknowledge acquired earlier.

Automating metadata acquisition

In Chapter 8 we have discussed some conditions that need to be satisfied so that we would have a competent AutoML/metalearning system. One of those conditions involves the size of metadata, which determines the quality of acquired metaknowledge. So the question is whether one should keep exploring the domain by conducting new tests or just focus on exploitation of the existing metaknowledge. Chapter 8 (Section 8.9), which discusses some strategies used in the area of multi-armed bandits, offers some suggestions regarding this problem. Still, more work is needed to provide the best possible solution to the issue.

18.3.4 Automating the reduction of configuration spaces

Many systems may adopt a strategy of defining a rather large configuration space in the search for a solution. This is motivated by the fact that, if the space was too small, the system would not be able to find a satisfactory model. However, configuration spaces that are too large also have disadvantages. As there are too many options to explore, the system may require a very long time to find the potentially best solution. Additionally, many alternatives may be redundant (i.e., lead to the same or a very similar model), which do not add value and waste resources. Chapter 8 discusses some strategies that can be used to reduce these spaces. The configuration spaces are affected by:

- The contents of a given portfolio with base-level algorithms
- The hyperparameters of algorithms and their possible values
- Ontologies/grammars/operators determining the space of admissible workflows

Each of these cases is analyzed next in a separate subsection.

Automating the reduction of base-level algorithms

One solution to the problem of reduction of base-level algorithms, based on metaknowledge acquired on prior tasks, was described in Chapter 8 (Section 8.5). The method included two basic steps: identify competent algorithms and eliminate redundant algorithms. It is conceivable that the proposed method can be further improved.

We note that even configuration metaknowledge can also be revised in an automatic way. This process can be seen as *meta-metalearning*, as its aim is to revise the existing metaknowledge.

Automating the reduction of hyperparameter spaces

Chapter 8 (Section 8.4) discusses methods that enable to identify the most important hyperparameters of a given algorithm. This is useful, as it enables the human designer to redefine the configuration space accordingly. However, the challenge is to design a system that would do this in an automatic way. One step in this direction are surrogate models that transfer knowledge about the behavior of parts of the space of hyperparameters across models (see Chapter 6).

Automating the reduction of workflow (pipeline) spaces

The given ontologies/grammars/operators can be regarded as another form of configuration metaknowledge, determining which workflows (pipelines) are admissible in the configuration space. The challenge is how to design a system that would be capable of learning/updating this type of metaknowledge automatically, for instance, from given examples of admissible/inadmissible workflows.

18.3.5 Automating data stream mining

Many real-world datasets are in fact continuous streams of data. Various approaches that employ metalearning were discussed in Chapter 11, which covered several approaches:

Several methods split the stream into various intervals of equal size and extract metafeatures per interval. It is possible to build a meta-model based on such features and predict what would be the potentially best algorithm (e.g., classifier) for the next interval.

Alternatively, BLAST trains multiple classifiers over the course of the data stream, and selects an appropriate classifier for a next set of observations, based on the performance on previous observations.

Finally, streams can contain seasonality and recurring concepts. By storing previous models, ensemble and/or metalearning approaches can be used to select an appropriate method for a next part of the stream.

Despite the variety of approaches, various challenges remain. One of the key aspects of the data stream setting is the occurrence of concept drift. At any moment in time, a gradual or abrupt change in the data can degrade the performance of earlier trained models. While drift detectors have served to overcome this difficulty, new work on self-assessing machine learning tools can play an important role in improving this aspect (König et al., 2020).

18.3.6 Automating neural network parameter configuration

Training deep neural networks is known to be both a data-intensive and time-consuming process. Modern metalearning approaches can help to solve these issues by transferring knowledge from related tasks. For many approaches, this knowledge transfer is on the level of parameters, rather than hyperparameters. *Few-shot learning* aims to apply these techniques to scenarios where only very few data items are available.

As these techniques are applicable to relatively homogeneous data sources, so a question arises regarding how to estimate the “degree of homogeneity”, as this could determine how much training is needed beyond few examples that would normally be used. As such, an important challenge is determining which datasets can be considered as “similar”, and developing techniques that automatically detect from which datasets knowledge can be transferred.

18.3.7 Automating of data science

The area of data science has attracted a lot of attention recently, as it represents a more general framework than, for instance, data mining. Consequently, the question of whether it is possible to use automated methods arose and was addressed by some researchers. In Chapter 14 we have presented our perspective of what data science should involve. In this chapter we have considered the following stages:

1. Defining the current problem/task
2. Identifying the appropriate domain-specific knowledge
3. Obtaining the data
4. Automating data preprocessing and transformation
5. Changing the granularity of representation
6. Searching for the best workflow
7. Automating report generation

Although every stage has its challenges, the work on some stages has advanced more than on others. For instance, various papers discuss methods appropriate for stage 4, involving automation of data preprocessing and transformation. The reader can consult Chapter 14 (Section 14.3) to obtain more details on this issue.

Stage 6, which is concerned with the search for the potentially best algorithm/configuration/workflow, was discussed in various chapters of this book. Stage 7 was briefly discussed in Chapter 14. It can be expected that the existing work provides a good basis for further progress.

In our view, stages 1, 2, 3, and 5 represent a more serious challenge, as not much work has been done that could aid automation. In the following subsections these stages are addressed in more detail.

Definition of the current problem/task

In Chapter 14 we argued that, although the initial description needs to be provided by the domain expert (e.g., in natural language), further processing can be done with the help of (semi-)automatic methods. We have presented some initial ideas on this matter, which involve the usage of “task descriptors (keywords)”, that can be used in further processing, such as the one discussed in the next subsection.

Identifying the appropriate domain specific metaknowledge

This issue is relevant for any system that aspires to resolve rather diverse problems. Here, the domain-specific knowledge involves metadata, metaknowledge acquired on prior tasks, and definition of appropriate configuration spaces (configuration metaknowledge). This has the advantage that it enables to focus on a smaller configuration space, and consequently the search for the potentially best solution is easier.

Obtaining the data

This step may be easy to carry out, provided someone has already told the system where the data is. Typically the data would be stored and retrieved from some kind of database or OLAP. This stage may, however, present challenges if we want the system to be more independent and not always rely on human assistance. As pointed out in Section 14.2, we may encounter a problem for which hardly any data is available. A great deal of human scientific work is of this kind. The scientist has to elaborate a plan, which normally involves some form of interaction with the outside world, or appropriate sites on the internet, to obtain the required data. In Section 14.2, we mention, for instance, the system *Robot Scientist* that does just this. Some data science systems of the next generation may want to include this facet in their design.

Changing the granularity of representation

This area is also not without challenges. As was pointed out in Chapter 14, many practical applications exploit the information in databases or in an OLAP cube to generate appropriate *aggregate data* so as to be able to advance with data mining. Although some papers addressed this issue in the past and the topic is of utmost importance in many practical applications, this did not seem to have been reflected much at some recent data science workshops (e.g., ADS 2019 mentioned before).

However, as pointed out in Chapter 15, other changes of granularity can be considered, apart from the aggregation mentioned above. So the challenge is whether these processes could be incorporated into a more advanced AutoDS system.

18.3.8 Automating the design of solutions with more complex structures

In Chapter 15 we discuss various future lines of research. It is pointed out that not all solutions can be represented in the form of workflows, as more complex structures may be required (e.g., conditional operators, iteration, etc.). So the challenge is how to extend the existing metalearning/AutoML approaches to enable solutions with more complex structures.

18.3.9 Designing metalearning/AutoML platforms

Various chapters of this book not only discussed different approaches but also provided details about the corresponding implemented systems. Particular attention was given to existing metalearning/AutoML platforms, as these can typically be applied to many diverse problems. So the platforms are very important not only for future research, but also for real-world deployment.

However, the platforms do not always include the latest advances in the area. So the challenge is to extend the existing platforms, design new ones, and conduct comparative studies that enable to identify the more competitive variants for further work.

Final Challenge to the Reader

In the last couple of years, we have seen many new solutions to both old and new problems, and many new challenges arose in this context. With the rise of AutoML and metalearning for deep neural networks, new research communities have emerged. We hope that we have gathered some interest in the reader that would inspire him/her to participate in the development of respective solutions. We plan to review these challenges and their solutions in the next edition of the book.

References

König, M., Hoos, H. H., and van Rijn, J. N. (2020). Towards algorithm-agnostic uncertainty estimation: Predicting classification error in an automated machine learning setting. In *7th ICML Workshop on Automated Machine Learning (AutoML)*.