SECOND PLANNING TO LEARN WORKSHOP (PLANLEARN) AT THE ICML/COLT/UAI 2008

PlanLearn-08

July 9, 2008

Helsinki, Finland

Editors:

Pavel Brazdil, University of Porto

Abraham Bernstein, University of Zurich

Larry Hunter, Univ. of Colorado at Denver and Health Sciences Center, USA

Typesetting:

Rui Leite, University of Porto

Preface

The task of constructing composite systems, that is systems composed of more than one part, can be seen as interdisciplinary area which builds on expertise in different domains. The aim of this workshop is to explore the possibilities of constructing such systems with the aid of Machine Learning and exploiting the know-how of Data Mining. One way of producing composite systems is by inducing the constituents and then by putting the individual parts together.

For instance, a text extraction system may be composed of various subsystems, some oriented towards tagging, morphosyntactic analysis or word sense disambiguation. This may be followed by selection of informative attributes and finally generation of the system for the extraction of the relevant information. Machine Learning techniques may be employed in various stages of this process.

The problem of constructing complex systems can thus be seen as a problem of planning to resolve multiple (possibly interacting) tasks. So, one important issue that needs to be addressed is how these multiple learning processes can be coordinated. Each task is resolved using certain ordering of operations. Meta-learning can be useful in this process. It can help us to retrieve previous solutions conceived in the past and re-use them in new settings.

The aim of the workshop is to explore the possibilities of this new area, offer a forum for exchanging ideas and experience concerning the state-of-the art, permit to bring in knowledge gathered in different but related and relevant areas and outline new directions for research.

Of particular interest are methods and proposals that address the following issues:

- Planning to construct composite systems,
- Exploitation of ontologies of tasks and methods,
- Representation of learning goals and states in learning,
- Control and coordination of learning processes,
- Recovering / adapting sequences of DM operations,
- Meta-learning and exploitation of meta-knowledge,
- Layered learning,
- Multi-task learning,
- Transfer learning,
- Multi-predicate learning (and other relevant ILP methods),
- Combining induction and abduction,
- Multi-strategy learning,
- Learning to learn.

Other areas may be covered, provided they are relevant towards the overall aims of the workshop.

Helsinki, July 2008

Pavel Brazdil
Abraham Bernstein
Larry Hunter

Workshop Organization

Workshop Chairs

Pavel Brazdil (University of Porto)
Abraham Bernstein (University of Zurich)
Larry Hunter (University of Colorado at Denver and Health Sciences Center, USA)

ICML Workshop Co-Chairs

Sanjoy Dasgupta (University of California, San Diego)
Michael Littman (Rutgers University)

Workshop Program Committee

Pavel Brazdil, LIAAD, University of Porto, Portugal

Christophe Giraud-Carrier, Brigham Young University, USA

Saso Dzeroski, IJS, Ljubljana

Peter Flach, Univ. of Bristol, Great Britain

Larry Hunter, Univ. of Colorado at Denver and Health Sciences Center, USA

Rui Leite, LIAAD, University of Porto, Portugal

Tom Mitchell, Carnegie-Mellon Univ., USA (advisory role)

Oliver Ray, Univ. of Bristol, Great Britain

Ashwin Ram, Georgia Tech, USA

Luc de Raedt, University of Leuven, Belgium

Carlos Soares, LIAAD, University of Porto, Portugal

Maarten van Someren, University of Amsterdam, The Netherlands

Vojtech Svatek, Univ. of Economics, Prague, Czech Republic

Ricardo Vilalta, University of Houston, USA

Filip Železný, Czech Technical University, Czech Republic

Contents

Invited Talks

Main Contributions

Posters

Invited Talks

New Directions in Goal-Driven Learning

Ashwin Ram ASHWIN@CC.GATECH.EDU

Cognitive Computing Lab, College of Computing, Georgia Tech, USA

Abstract

Goal-Driven Learning (GDL) views learning as a strategic process in which the learner attempts to identify and satisfy its learning needs in the context of its tasks and goals. This is modeled as a planful process where the learner analyzes its reasoning traces to identify learning goals, and composes a set of learning strategies (modeled as planning operators) into a plan to learn by satisfying those learning goals.

Traditional GDL frameworks were based on traditional planners. However, modern AI systems often deal with real-time scenarios where learning and performance happen in a reactive real-time fashion, or are composed of multiple agents that use different learning and reasoning paradigms. In this talk, I will discuss new GDL frameworks that handle such problems, incorporating reactive and multi-agent planning techniques in order to manage learning in these kinds of AI systems.

Short C.V.

Dr. Ashwin Ram is an Associate Professor and Director of the Cognitive Computing Lab in the College of Computing at Georgia Tech, an Associate Professor of Cognitive Science, and an Adjunct Professor in Psychology at Georgia Tech and in MathCS at Emory University. He received his PhD from Yale University in 1989, his MS from University of Illinois in 1984, and his BTech from IIT Delhi in 1982. He has published 2 books and over 100 scientific articles in international forums. He is a founder of Enkia Corporation which provides AI software for information assurance and decision support.

Transfer Learning by Mapping and Revising Relational Knowledge

Raymond J. Mooney MOONEY@CS.UTEXAS.EDU

University of Texas at Austin

Abstract

Transfer Learning (TL) attempts to leverage knowledge previously acquired in a source domain to improve the accuracy and speed of learning in a related target domain. Statistical Relational Learning (SRL) concerns methods that combine the strengths of predicate logic and probabilistic graphical models in order to effectively and robustly learn and reason about complex relational data. Our recent work uses TL to improve SRL, specifically transfering learned Markov Logic Networks (MLNs), an expressive SRL formalism, to new domains. We present a complete MLN transfer system that first autonomously maps the predicates in the source MLN to the target domain and then revises the mapped knowledge to further improve its accuracy. Experimental results in several real-world domains demonstrate that our approach successfully reduces the amount of time and training data needed to learn an accurate model of a target domain over learning from scratch. A future research issue that concerns planning to learn is automatically selecting useful source domains for a given target domain or actually constructing simpler problems to use as potential sources for transfer.

Main Contributions

Experiment Databases: Creating a New Platform for Meta-Learning Research

Keywords: meta-learning, databases, model selection, ontologies

Joaquin Vanschoren JOAQUIN.VANSCHOREN@CS.KULEUVEN.BE
Hendrik Blockeel HENDRIK.BLOCKEEL@CS.KULEUVEN.BE
Department of Computer Science, K.U.Leuven, Leuven, Belgium

Bernhard Pfahringer BERNHARD@CS.WAIKATO.AC.NZ
Geoff Holmes GEOFF@CS.WAIKATO.AC.NZ
Department of Computer Science, University of Waikato, Hamilton, New Zealand

Abstract

Many studies in machine learning try to investigate what makes an algorithm succeed or fail on certain datasets. However, the field is still evolving relatively quickly, and new algorithms, preprocessing methods, learning tasks and evaluation procedures continue to emerge in the literature. Thus, it is impossible for a single study to cover this expanding space of learning approaches. In this paper, we propose a community-based approach for the analysis of learning algorithms, driven by sharing meta-data from previous experiments in a uniform way. We illustrate how organizing this information in a central database can create a practical public platform for any kind of exploitation of meta-knowledge, allowing effective reuse of previous experimentation and targeted analysis of the collected results.

1. Introduction

1.1. Sharing Meta-Data

Research in machine learning is inherently empirical. Researchers, as well as practitioners, seek a deeper understanding of learning algorithm performance by performing large numbers of learning experiments. Whether the goal is to develop better learning algorithms or to select useful approaches to analyze new sources of data, collecting the right meta-data and correctly interpreting it is crucial for a thorough understanding of learning processes.

Despite an abundance of empirical studies (and meta-data), much remains to be learned about what makes an algorithm succeed or fail on certain datasets. Several comprehensive empirical studies, such as StatLog (Michie et al., 1994), MetaL (Brazdil et al., 2003) and, more recently, Ali and Smith (2006) and Caruana and Niculescu (2006) try to provide an overview of the state-of-the-art, but as new algorithms, preprocessing methods, learning tasks, and evaluation metrics are introduced at a constant rate, it is impossible for a single study to cover this continuously expanding space of learning approaches. Moreover, the meta-data generated by these and thousands of other machine learning studies is usually collected and stored differently and therefore hard to share and reuse. Collecting this information in public repositories would create a resource that could be tapped at any time to retrieve up-to-date results from a wide range of prior studies.

Furthermore, especially from a practitioner's perspective, data mining is not a one-shot operation. Generally, many different preprocessing and modeling techniques have to be applied in order to gain a deeper understanding of the data at hand, and sharing meta-data about previous studies could greatly help practitioners to build on previous experience and check which methods might be particularly useful.

1.2. A Community-Based Approach

In this paper, we propose a community-based approach for the analysis of learning algorithms, driven by collecting and sharing meta-data about learning

Appearing in *Proceedings of the ICML/COLT/UAI 2008 Planning to Learn Workshop (PlanLearn)*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

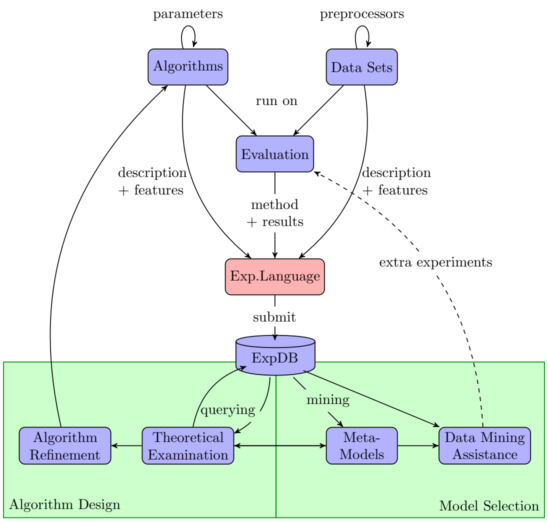

Figure 1. Using experiment databases[1]

methods, preprocessing methods and datasets in a uniform way, as illustrated in Fig. 1.

To share the results of any empirical machine learning study, they are first expressed in a standard experiment description language, capturing the details of the algorithms used, their parameter settings, the datasets on which they were run, the employed preprocessing steps, the evaluation methodology used and the results of that evaluation. Known features of the algorithms (e.g. its bias-variance profile) and datasets (e.g. landmarker evaluations) can also be added at any time. Next, the experiments are submitted to a repository, where they are automatically organized to make this information easily accessible and searchable. To this effect, each experiment is associated with previously stored experiments and linked to all known properties of the algorithms and datasets.

There are various ways to make use of such a resource. When evaluating or designing new methods, the repository could be queried to compare different algorithms or to test specific aspects of an algorithm's behavior, for instance how data properties or parameter settings affect its performance. The returned results can then be theoretically interpreted, possibly leading to new insights and/or improved algorithms. Furthermore, given the amount of available meta-data, it is also possible to mine the repository for patterns in algorithms behavior, or to use the shared data in data mining assistance tools to propose interesting approaches to new problems, possibly setting up more experiments to ascertain their validity.

The remainder of this paper is organized as follows. In Section 2, we discuss how experiment databases can presently serve as a useful repository for machine learning information. In Section 3, we show how to use such databases to answer various interesting research questions about the behavior of learning algorithms, and in Section 4 we discuss meta-models generated by mining the data. Finally, we consider some interesting future applications in Section 5, including possible uses in data mining assistance tools. Section 6 concludes.

2. Experiment Databases

An experiment database, as described in (Blockeel & Vanschoren, 2007), is a database specifically designed to store learning experiments in an organized fashion, including all details about the algorithms used, parameter settings, datasets, preprocessing methods, evaluation procedure and results. Once stored, all information can be easily accessed by writing the right database query, in standard SQL, providing a very versatile means to investigate large amounts of experimental results, both under very specific and very general conditions. With all details publicly available, they also ensure experiments can be easily reproduced and reused in future analysis.

Our implementation of such a database currently contains about 500,000 experiments of 54 classification algorithms on 87 datasets with 50 dataset characterizations, each evaluated on 36 evaluation metrics. It also contains some characterizations of the algorithms, like the model type, a bias-variance profile, and their compatibility with different types of data. It is available online at `http://expdb.cs.kuleuven.be`. This website also hosts a description of a standard experiment description language, available tools for uploading experiments, a gallery of SQL queries (including the ones used in the next section), and a query interface including visualization tools for displaying returned results.

3. Querying for Answers

To illustrate the use of such databases in various stages of the knowledge discovery process, we try a number of example queries, increasingly making use of the available meta-level descriptions of algorithms and datasets. While the first queries only use the recorded performance evaluations of specific algorithms on specific datasets, subsequent queries also use the stored data characteristics and algorithm features, offering increasingly generalizable results. A wider range of interesting queries can be found in Vanschoren et al. (2008).

[1] Figure adapted from a framework by Smith (2008).

3.1. Ranking Algorithms

When selecting interesting learning approaches, or when testing the effectiveness of a new algorithm, one is often interested in a ranking of the available methods. For instance, to investigate whether some algorithms consistently rank high over various problems, we can query for their average rank (using each algorithm's optimal observed performance) over a large number of datasets. Fig. 2 shows the result of a single query over all UCI datasets[2]. To check which algorithms perform significantly different, we used the Friedman test, as discussed in Demšar (2006). The right axis shows the average rank divided by the *critical difference*, meaning that two algorithms perform significantly different if the average ranks of two algorithms differ by at least that value (one unit)[3].

This immediately shows that indeed, some algorithms rank much higher on average than others on the UCI datasets. Boosting and Bagging, if used with the correct base-learners, perform significantly better than SMO (an SVM trainer), and SMO in turn performs better than J48 (C4.5). We cannot yet say that SMO is also better than the MultilayerPerceptron or RandomForests: more datasets (or fewer datasets in the comparison) are needed to reduce the critical difference. Note that the average rank of Bagging and Boosting is close to two, suggesting that a (theoretical) meta-algorithm that reliably chooses between the two approaches (and the underlying base-learner) would yield a very high rank[4].

Of course, to be fair, we should differentiate between different base-learners and kernels. We can drill down through the previous results by adjusting the query, additionally asking for the base-learners and kernels involved, yielding Fig. 3. Bagged naive Bayes trees seem to come in first overall, but the difference is not significant compared to that of SMO with a polynomial kernel (although it is compared to the RBF kernel). Also note that bagging and boosting PART and NBTrees seem to yield big performance boosts, while boosting random trees proves particularly ineffective.

Many more types of rankings can be generated with similar queries. For instance, if datasets are labeled as belonging to a specific task (e.g. image recognition), more specific queries could rank all algorithms for that particular task.

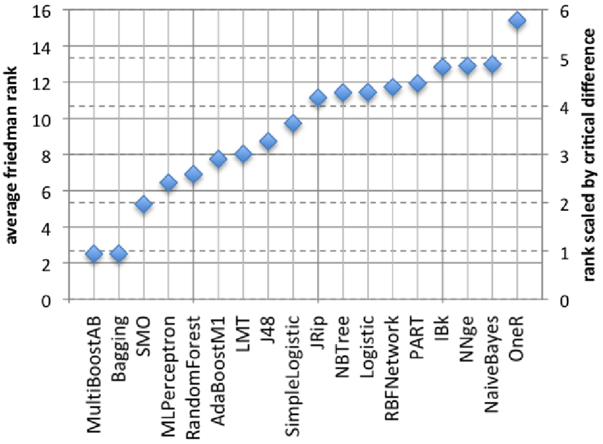

Figure 2. Algorithm ranking over all UCI datasets.

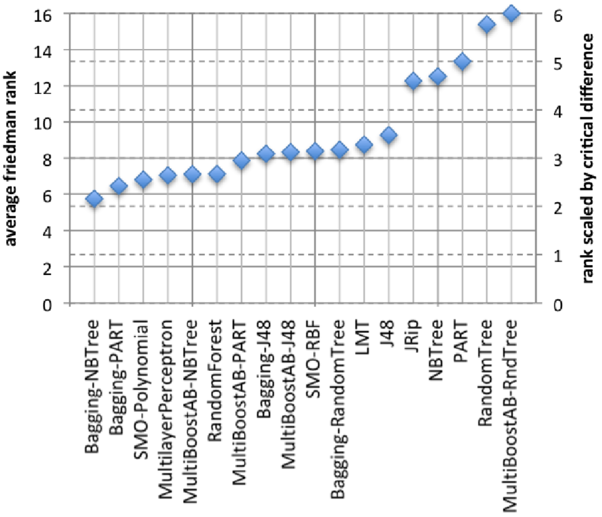

Figure 3. Algorithm ranking over all UCI datasets, including base-learners and kernels.

3.2. Tuning Parameters

Next to performance, there may be other reasons for selecting a certain algorithm. In that case, one still needs to tune the method's parameters to fit the given data. To learn from previously stored data, one could query for the effect of those parameters on other datasets. For instance, Fig. 4 shows the effect of the gamma parameter of the RBF kernel on a number of different datasets. Note that on some datasets, increasing the gamma value will steeply increase performance, until it reaches a maximum and then slowly declines, while on other datasets, performance immediately starts decreasing up to a point, after which it quickly drops to default accuracy. Querying for the number of attributes in each dataset (shown in brack-

[2] We selected 18 algorithms to limit the amount of statistical error generated by using 'only' 87 datasets

[3] The critical difference was calculated using the Nemenyi test with p=0.1, 18 algorithms and 87 datasets

[4] Rerunning the query while joining Bagging and Boosting yields an average rank of 1.7, down from 2.5.

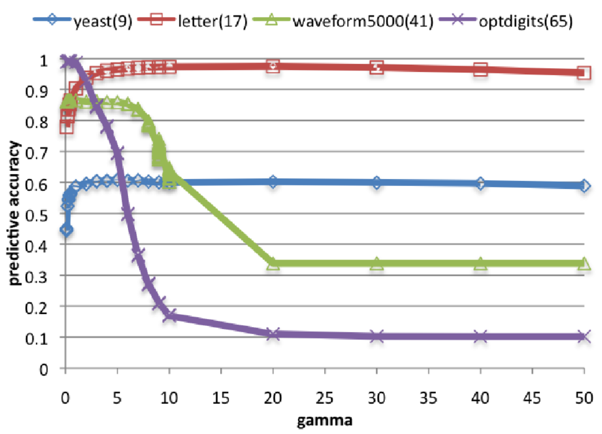

Figure 4. Effect of the RBF-kernel's gamma parameter on different datasets.

ets), suggests that only datasets with few attributes benefit from large gamma values[5]. This is one example where a query suggests ways to improve an algorithm: by taking the number of attributes into account, the kernel can be made less sensitive to this characteristic.

3.3. Applying Preprocessors

In many cases, data needs to be preprocessed before a certain learning algorithm can be used effectively. Since the database can, to a certain extent, also store preprocessing methods, we can investigate their effect on the performance of learning algorithms. For instance, to investigate how much data a certain algorithm needs to perform optimally, we can query for the results on downsampled versions of the dataset, yielding a learning curve for each algorithm, as shown in Fig. 5. Note that the learning curves cross, indicating that the ranking of algorithms depends on the size of the sample. While logistic regression is initially stronger than J48, the latter keeps on improving when given more data[6]. Also note that RandomForest is consequently better, improving steadily when given more data, that RacedIncrementalLogitBoost (with decision stumps as its base-learner) has a particularly steep learning curve, crossing two other curves, and that the performance of the HyperPipes algorithm actually worsens given more data.

However, to be truly useful for investigating the effect of preprocessing methods, the database model will need to be extended further, as discussed in Section 5.

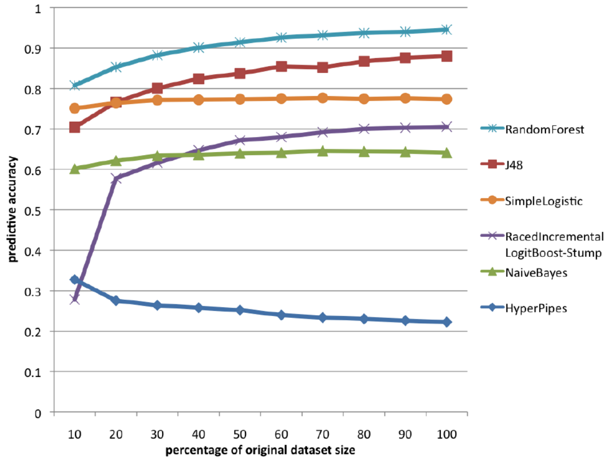

Figure 5. Learning curves on the Letter-dataset.

3.4. Generalizing over Algorithms

Instead of querying for a specific algorithm to use on certain types of data, we may also want to investigate which general properties of an algorithm might be desirable when approaching a certain task. One very interesting property of an algorithm is its bias-variance profile (Kalousis & Hilario, 2000). Since the database contains a large number of bias-variance decomposition experiments[7], we can give a realistic, numerical assessment of how capable each algorithm is in reducing bias and variance error. The average percentage of bias-related error, compared to the total error, is stored in the database as an algorithm property.

In a final query, we investigate the claim by Brain & Webb (2002) that on large datasets, the bias-component of the error becomes the most important factor, and that we should use algorithms with high bias management to tackle them. To verify this, we look for a connection between the dataset size and the proportion of bias error in the total error of a number of algorithms, selecting algorithms with very different bias-variance profiles. Averaging the bias-variance results over datasets of similar size for each algorithm produces the result shown in Fig. 6. It shows that bias error is of varying significance on small datasets, but steadily increases in importance on larger datasets, for all algorithms. This validates the previous study on a larger set of datasets. In this case (on UCI datasets), bias becomes the most important factor on datasets larger than 50000 examples, no matter which algorithm is used. As such, it is indeed advisable to look to algorithms with good bias management when dealing with large datasets.

[5] This relationship is investigated in more detail in Vanschoren et al. (2008).

[6] This confirms earlier analysis by Perlich et al. (2003), even though in that study, the dataset was transformed to a binary problem.

[7] The database stores both Kohavi-Wolpert's and Webb's definition of bias/variance. We use the former.

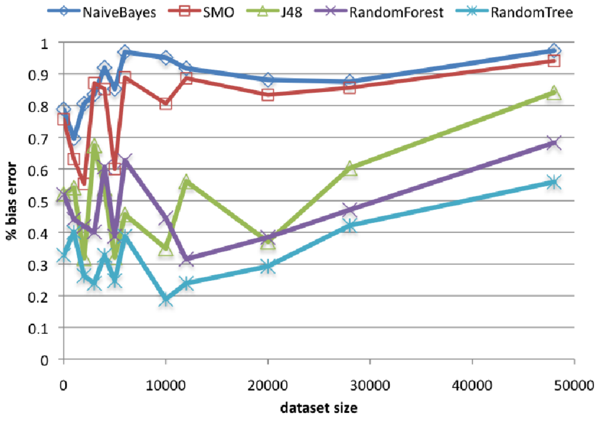

Figure 6. The average percentage of bias-related error in algorithms as a function of dataset size.

4. Mining for Patterns

Instead of studying different dataset properties independently, we could also use data mining techniques to relate the effect of many different properties to an algorithm's performance. For instance, when looking at Fig. 2, we see that OneR ranks, on average, much lower than the other algorithms, while earlier studies, most notably one by Holte (1993), found very little performance difference between OneR and the more complex J48. In fact, when querying for the default performance of OneR and J48 on all UCI datasets, and plotting them against each other, we get Fig. 7, showing that on a large number of datasets, their performance is indeed about the same (their performances cross near the diagonal). Still, J48 works much better on many other datasets.

To automatically learn under which conditions J48 clearly outperforms OneR, we queried for the characteristics of each dataset, and discretized the data into three class values: "draw", "win_J48" (>4% gain), and "large_win_J48" (>20% gain). The tree returned by J48 on this meta-dataset is shown in Fig. 8, showing that a high number of class values often leads to a large win of J48 over 1R.

Probably, many more meta-models remain to be discovered by simply querying the database and and mining the results.

5. Future Work

Although the system discussed above readily allows users to share and explore machine learning meta-data, there are several opportunities for further development.

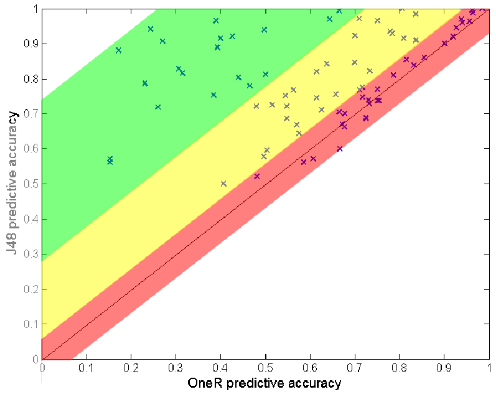

Figure 7. Performance comparison of J48 and OneR on all UCI datasets, discretized in 3 classes: draw (red), win_J48 (yellow), large_win_J48 (green).

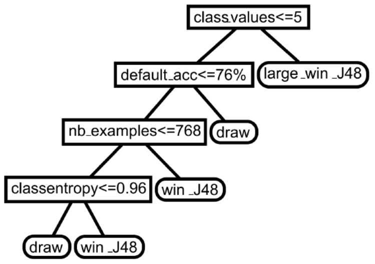

Figure 8. Meta-decision tree showing on which data J48 is better than OneR

First of all, the current support for preprocessing methods is rather limited as the database can currently only store sequences of preprocessing steps. In reality, complex workflows are used to prepare the data before modeling, and the database model should be extended to capture such workflows. Unfortunately, there is, to our knowledge, no standard language to describe KDD workflows. Still, this is a challenge, not a limitation. One interesting approach using ontologies to describe preprocessing steps in bio-informatics can be found in Pérez-Rey et al. (2006).

Secondly, next to querying and mining the repository, it would be interesting to use the available meta-data and querying possibilities in data mining assistance tools. A number of such tools have been proposed before, some of which could be enhanced by making use of the repository. One very interesting approach is the Intelligent Discovery Assistant proposed by Bernstein et al. (2005), which defines an ontology for the data mining process and uses it to generate a ranking of interesting approaches to handle a certain data mining task. However, the effects and properties of preprocessors and algorithms are hard-coded. Using an experiment repository could engender a much more flexible system, querying the database to see, for instance, how fast an algorithm would be on a certain dataset, or what the preconditions are for applying newly entered algorithms.

New installments of algorithm ranking tools, like the MetaL ranker (Brazdil et al., 2003) can also be envisaged. For instance, such a ranker would be able to use any stored evaluation metric, or a combination thereof, take parameter variations into account and include viable preprocessing steps. Furthermore, it could be given a certain amount of time to optimize its predictions by automatically running additional experiments.

6. Conclusions

We propose a community-based approach for the analysis of learning algorithms in which both machine learning researchers and practitioners can share meta-data from learning experiments in a uniform way, and upload this information to a searchable experiment database. We illustrate the use of such databases in various stages of the knowledge discovery process by discussing a number of example queries, increasingly making use of the available meta-level descriptions of algorithms and datasets. These include ranking all algorithms on a group of datasets, comparing parameter effects on different types of datasets, using preprocessors to draw learning curves, and plotting the average bias-variance ratio of several algorithms against the dataset size, each in a single query. Next, we illustrated how the available meta-data can also be automatically mined for patterns in algorithm behavior. Finally, we discussed possibilities for future work, including applications in data mining assistance tools.

Acknowledgements

Hendrik Blockeel is Postdoctoral Fellow of the Fund for Scientific Research - Flanders (Belgium) (F.W.O.-Vlaanderen), and this research is further supported by GOA 2003/08 “Inductive Knowledge Bases”.

References

Ali, S., & Smith, K. A. (2006). On learning algorithm selection for classification. *Applied Soft Computing*, *6*, 119–138.

Bernstein, A., Provost, F., & Hill, S. (2005). Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering*, *17*, 503–518.

Blockeel, H., & Vanschoren, J. (2007). Experiment databases: Towards an improved experimental methodology in machine learning. *Lecture Notes in Computer Science 4702: PKDD 2007 Proceedings* (pp. 6–17). Springer.

Brain, D., & Webb, G. (2002). The need for low bias algorithms in classification learning from large data sets. *Lecture Notes in Computer Science 2431: PKDD 2002 Proceedings* (pp. 62–73).

Brazdil, P., Soares, C., & Pinto da Costa, J. (2003). Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning*, *50*, 251–277.

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. *ICML ’06: Proceedings of the 23rd international conference on Machine learning* (pp. 161–168). Pittsburgh, Pennsylvania.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J.Mach.Learn.Res.*, *7*, 1–30.

Holte, R. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, *11*, 63–91.

Kalousis, A., & Hilario, M. (2000). Building algorithm profiles for prior model selection in knowledge discovery systems. *Engineering Intell. Syst.*, *8(2)*.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine learning, neural and statistical classification.* Ellis Horwood.

Pérez-Rey, D., Anguita, A., & Crespo, J. (2006). Ontodataclean: Ontology-based integration and preprocessing of distributed data. *ISBMDA* (pp. 262–272).

Perlich, C., Provost, F., & Simonoff, J. S. (2003). Tree induction vs. logistic regression: a learning-curve analysis. *J. Mach. Learn. Res.*, *4*, 211–255.

Smith-Miles, K. (2008). Cross-disciplinary perspectives on the algorithm selection problem: recent advances and emerging opportunities. *ACM Computing Surveys*, to appear.

Vanschoren, J., Pfahringer, B., & Holmes, G. (2008). *Learning from the past with experiment databases.* (Technical Report Working Paper Series 08/2008). Computer Science Department, University of Waikato.

Learning to Plan: Predicting the Behavior of Two Heuristics on the Job-Shop Scheduling Problem

Pedro Abreu PEDABREU@LIAAD.UP.PT
LIAAD-INESC Porto LA, Rua de Ceuta 118 6o andar, 4050-190 Porto, Portugal

Carlos Soares CSOARES@FEP.UP.PT
LIAAD-INESC Porto LA
Faculdade de Economia, Universidade do Porto

Jorge M. S. Valente JVALENTE@FEP.UP.PT
Faculdade de Economia, Universidade do Porto

Abstract

We present a general methodology to model the behavior of heuristics for the Job-Shop Scheduling that address the problem by solving conflicts between different operations on the same machine. Our approach is based on the prediction of the gaps in the operation of machines, which are then combined to estimate the performance of the methods. We tested it using two well know heuristics: Shortest Processing Time and Longest Processing Time. Our results show that it is possible to predict the behavior of these heuristics on random instances, generated using different distributions. We have also the same models to estimate the relative performance of two heuristics, but without success in this case.

1. Introduction

The complexity of optimization problems such as the Job-Shop Scheduling Problem (JSS) makes it very difficult to understand the behavior of heuristic methods. This is true even for simple ones, such as the Shortest Processing Time (Jain & Meeran, 1999). An interesting research question is whether it is possible to create models that relate properties of JSS instances with the performance of different heuristics. In this paper, we address this problem using a Machine Learning approach.

If we are able to successfully generate such models, they will be useful for several purposes. They can improve our understanding of the JSS problem and the methods used to solve it. Additionally, they can provide insights on how to improve those methods or develop new ones. Finally, given an instance of the JSS problem, they can help to decide which method should be used.

Two very simple and common heuristics to solve the JSS problem are considered here, namely the Shortest Processing Time (SPT) and the Longest Processing Time (LPT) methods (Jain & Meeran, 1999).

In Section 2 we described the background for this work and provide further motivation. Our approach is described in Section 3 and the results obtained are presented in Section 4. We analyze the results and present our conclusions in Section 5.

2. The Job-Shop Scheduling Problem

The deterministic job-shop scheduling problem can be seen as the most general of the classical scheduling problems. Formally, this problem can be described as follows. A finite set J of n jobs $\{J_1, J_2, \ldots, J_n\}$ has to be processed on a finite set M of m machines $\{M_1, M_2, \ldots, M_m\}$. Each job J_i must be processed once on every machine M_j, so each job consists of a chain of m operations. Let O_{ij} represent the operation of job J_i on machine M_j, and let p_{ij} be the processing time required by operation O_{ij}.

The operations of each job J_i have to be scheduled in a predetermined given order, i.e. there are precedence constraints between the operations of each job J_i. Let $\prec$ be used to denote a precedence constraint, so that

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

$O_{ik} \prec O_{il}$ means that job J_i has to be completely processed on machine M_k prior to being processed on machine M_l. Each job has its own flow pattern through the machines, so the precedence constraints between operations can be different for each job. Other additional constraints also have to be satisfied. Each machine can only process one job at a time (capacity constraints). Also, preemption is not allowed, so operations cannot be interrupted and must be fully processed once started. Let t_{ij} denote the starting time of operation O_{ij}. The objective is to determine starting times t_{ij} for all operations, in order to optimize some objective function, while satisfying the precedence, capacity and no-preemption constraints. The duration in which all operations for all jobs are completed is denoted as the makespan $C_{\max}$. In this paper, we consider as objective function the minimization of the makespan:

$$\begin{aligned} C^*_{\max} &= \min\left(C_{\max}\right) \\ &= \min_{\text{feasibleschedules}}\left(\max\left(t_{ij}+p_{ij}\right)\right), \\ &\quad \forall J_i \in J, M_j \in M. \end{aligned}$$

The job-shop scheduling problem is NP-hard (Garey & Johnson, 1979; Lenstra & Rinnooy Kan, 1979), and notoriously difficult to solve. Many papers have been published on the job-shop scheduling problem. A comprehensive survey of job shop scheduling techniques can be found in (Jain & Meeran, 1999). Given the complexity of the job-shop scheduling problem, the exact methods are limited to instances of small size. Metaheuristic algorithms have been successfully applied to instances of small to medium size. However, for large instances, dispatching rules are the only heuristic procedure that can provide a solution within reasonable computation times. Furthermore, dispatching rules are also often required for other heuristic procedures, e.g. metaheuristic algorithms frequently use dispatching rules to generate initial solutions.

The longest processing time (LPT) and shortest processing time (SPT) heuristics are two of the most well-known dispatching rules, and are widely used for the job-shop scheduling problem, as well as for a large number of other scheduling problems. In this paper, we consider these two rules, implemented with an active schedule generation algorithm (Giffler & Thompson, 1960; Baker, 1974). This algorithm assigns operations to machines as soon as possible, taking into account the constraints described earlier. Following this strategy, *conflicts* will probably occur. This means that two or more operation will overlap on a given machine if they are scheduled as soon as possible. In that case, the algorithm uses a rule to choose the order in which the operations will be assigned to the machine. Many rules can be used for that purpose. In this work, we use the dispatching rules LPT and SPT for that purpose. The LPT rule schedules the operation with the longest processing time and SPT chooses the operation with the shortest processing time. As illustrated in Figure 1, the performance of these heuristics differs across different instances. STP achieves the best result in this case as in 63.7% of the 1000 instances used in this work while LTP is the best in the 36.1%.

3. Learning the Behavior of Heuristics

Our goal is to use Machine Learning methods to induce models that relate the characteristics of JSS instances with the performance of these heuristics. One approach is to do this directly, using a learning algorithm to generate a mapping between the values of features representing those characteristics and the performance of each of the heuristics. Alternatively, one could try to predict the relative performnce of the heuristics, if the goal is to select the best one (Abreu & Soares, 2007). This assumes that it is

Here, we follow an alternative approach, because a schedule is the result of a a series of decisions made by an heuristic for every conflict that occurs during the construction of the schedule. Therefore, the makespan is the combination of the gaps between operations, as illustrated in Figure 1. Based on this assumption, our method is divided into two steps, which are discussed in the following sections:

1. Predict the gaps separating two consecutive operations of the same job, on the schedules generated by the heuristic;

2. Calculate the makespan, using the predicted gaps.

3.1. Prediction of the Individual Gaps

Our goal is to predict the gaps generated by the two heuristics considered, SPT and LPT, for each pair of consecutive operations, $O_{jm_{o-1}}$ and O_{jm_o}, defined as $Gap^{SPT}(O_{jm_o})$ and $Gap^{LPT}(O_{jm_o})$, respectively. The $Gap^{sched}_{jm_o} = t_{jm_o} - e_{jm_{o-1}}$ is the length of the period between two consecutive operations in job j in a given schedule $sched$, $e_{jm_o} = t_{jm_o} + p_{jm_o}$ is the end time of operation O_{jm}, m_o is the machine in precedence order $o \in \{2, \ldots, |M|\}$ and $Gap^{sched}_{jm_1} = t_{jm_1}$. The value of Gap is non-zero when the beginning of the second operation (O_{jm_o}) has been delayed by a conflict, which was solved by the heuristic in favour of another operation. In Figure 1 we observe two schedules generated by SPT and LPT, respectively. It illustrates how the

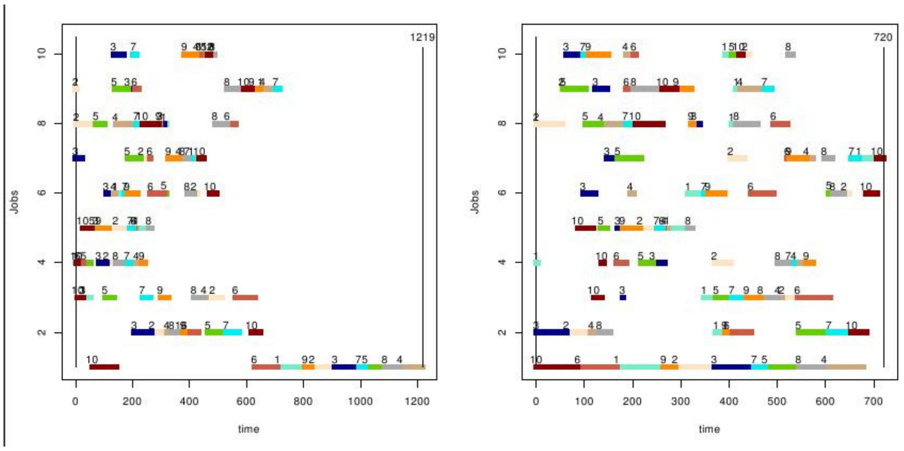

Figure 1. Schedules generated for the same instance with LTP (left) and STP (right).

use of different heuristics can yield large differences in gaps.

We address this as a regression problem (Kononenko & Kukar, 2007). In regression, the goal is to obtain a model that relates a set of independent variables, or features, X_i with a target variable, Y, based on a set of examples for which both the values of $\{X_1, X_2, \ldots\}$ and Y are known. In our case, an example is a conflict in a schedule and the target variable, Y, is the gaps, $Gap^{sched}(O_{jm_o})$.

3.2. Features to Describe Conflicts

In order to obtain a reliable model, regression methods must be provided with features, $\{X_1, X_2, \ldots\}$ which are good predictors of the target variable, Y. In other words, to be able to make accurate predictions, features that provide information about the target variable are required.For this problem, we need to design features representing operations (e.g., duration), usage of machines (e.g., total processing time required) and relations between operations in a job and using the same machine.

Some of the measures used here are based on the Infinite Capacity Schedule (ICS). The ICS is the schedule obtained by relaxing the capacity constraints (i.e., the constraints that specify that each machine can only process one job at a time). This schedule can be easily constructed by scheduling the operations as specified by the precedence constraints. Based on the ICS, we compute a measure of the distance between two operations ($O_{j_1 m}$ and $O_{j_2 m}$) that are processed in the same machine as follows: $dist(O_{j_1,m}, O_{j_2,m}) = \frac{|m_{j_1 m} - m_{j_2 m} + 1|}{\frac{p_{j_1 m} + p_{j_2 m}}{2} + 1}$, where $m_{jm} = t_{jm} + \frac{p_{jm}}{2}$ is the time unit when half of operation O_{jm} is processed. When the operations are consecutive, the value of the measure is 1 and when the two operations are centered on the same time unit, the value depends only on the duration of operation. This means that if two operations are near and have a long duration, the value of this measure is low, so there is a high possibility of having a conflict when trying to generate a feasible schedule. Some additional notation is: $d(S)$ is the set containing the duration of operations in S, $d(S) = \{p_{ij} : \forall O_{ij} \in S\}$; ld_{ij} (hd_{ij}) is the subset of M_j containing operations with shorter (longer) duration than O_{ij}, $ld_{ij} = \{O_{rj} \in M_j : p_{rj} < p_{ij}\}$ ($hd_{ij} = \{O_{rj} \in M_j : p_{rj} > p_{ij}\}$); $dist_{jm_o}(S)$ is the ICS-based distance measure between O_{jmo} and the other operations in set S, $dist_{jm_o}(S) = \{dist(O_{jm_o}, O_{j_1 m_o}) : \forall O_{j_1 m_o} \in S\}$; $dist_{ij}^{[s,r]}$ is the subset of M_j containing operations with distance of ICS to O_{ij} between s and r. Additionally, some of the measures are computed for several operations in a machine and aggregated using functions sum, average, minimum, maximum, median, variance, the first and the third quartile, represented below as f, for simplicity. Therefore, the features used to characterize a given operation (O_{jm_o}) are:

- **Precedence Order** The pre-defined order of the operation in the job, o
- **Duration Rank in Machine** The position of the operation in a ranking of the operations in the same machine in increasing order of duration, $|ld_{jm_o}| + 1$
- **Processing Time in Machine** Aggregated duration of the operations to be processed on the same machine as O_{jm_o}, $f(d(M_{m_o}))$
- **Distance to Other Operations in Machine**

Aggregated distance in the ICS between the operation and the other operations to be processed on the same machine, $f(dist_{jm_o}(M_{m_o} \backslash \{O_{jm_o}\}))$

- **Duration of Operations with Shorter/Longer Duration in Machine** Aggregated duration of operations to be processed on the same machine that have shorter/longer duration than operation O_{jm_o}, $f(d(ld_{jm_o}))/f(d(hd_{jm_o}))$
- **Distance to Other Operations in Machine with Shorter/Longer Duration** Aggregated distance in the ICS between the operation and the other operations with shorter/longer duration to be processed in the same machine, $f(dist_{jm_o}(ld_{jm_o}))/f(dist_{jm_o}(hd_{jm_o}))$
- **Duration with distance higher than r and less than s and Shorter/Longer duration** Aggregated duration of the operation with distance in the ICS to operation O_{jm_o} between $[r, s]$ and with duration shorter/longer than the operation O_{jm_o}, f(S) where $S = dist_{jm_o}^{[r,s]} \cap ld_{jm_o}(hd_{jm_o})$. The values of $[r, s]$ are:$] - \infty, -3],] - 3, -1.5],] - 1.5, 0],]0, 1.5],]1.5, 3]$ and $]3, \infty[$

The features are not the same for the two heuristics. For prediction of the gaps of the heuristic SPT, we use the features describe above using the Shorter characteristic (from shorter/Longer) and for the heuristic LPT, we use the Longer characteristic.

3.3. Predicting the Makespan

The predictions of the gaps (Gap) for each schedule can be used to estimate the makespan of the schedule generated using the rule R for instance I as follows

$$MK_I^R = max_{j \in \{1,\dots,|J|\}} \left(\sum_{o=1}^{|M|} \left(p_{jm_o} + Gap^R \left(O_{jm_o} \right) \right) \right)$$

Note that these estimates can be used to select the best heuristic. SPT should be used if $MK_I^{SPT} < MK_I^{LPT}$, otherwise LPT should be used.

4. Experiments

We have tested four regression methods that are available in the R statistical package (`www.r-project.org`): regression trees, linear regression, support vector machines (SVM) and random forest. For the linear model, we redimension the features In order to analyse the results, we consider the empirical plots of the real against the predicted data, and also use the following analytical error measure - relative mean squared error ($RMSE$). This error is calculated as $RMSE = \frac{\sum_i (f_i - \hat{f}_i)^2}{\sum_i (f_i - \bar{f}_i)^2}$, where $\hat{f}_i$ is the prediction of the target feature for example i from the test dataset, f_i is the real value of the target feature of example i from the test dataset and the $\bar{f}_i$(baseline) is the average of the values of the target feature on the training dataset. We note that if the value of RMSE is greater than 1, it means that the learning algorithm did not learn a model capable of generalizing to new examples better than by using the mean value of the target on the training set.

We generated three types of datasets, namely Uniform, Gaussian and Beta datasets. These datasets differ in the method used to generate the JSS instances that are tested. In the Uniform and Gaussian datasets, the precedence order is generated using a uniform distribution, as described in Taillard (Taillard, 1993). Also, in the Uniform (Gaussian) dataset, the duration of each operation is generated using a uniform (gaussian) distribution, such that these durations are not correlated with the machines or jobs, as described in (Watson et al., 1999). The Beta dataset contains more diverse instances, with duration of operations either not correlated with machines or jobs, correlated with machines or correlated with jobs, and randomly generated parameters α and β. For the precedence order in the instances of the Beta dataset, instead of generating a new precedence order for each job, as in the Uniform and Gaussian datasets, a previously generated precedence order is sometimes repeated. The larger the number of repetitions, the closer the instance is to a flowshop instance, since in the flowshop problem the precedence constraints are identical for all jobs. Each instance has 5 jobs to be processed on 5 machines and the range of processing time values is between 1 and 99. Experiments were carried out separately for the data corresponding to each of the distributions. We have generated 1000 instances using each of the distributions, which corresponds to 25000 examples, each one representing one gap of consecutive operations. For each experiment the data corresponding to 500 instances were used for training and the remaining data were used for testing. Note that, given the large size of the test set, there is no need to employ resampling techniques, such as cross-validation, for estimating the error;

In the figures, we have on the x-axis the prediction based on the predictions obtained with the Random Forest model and on the y-axis, we have the real value.

Table 1. Error results obtained with different models (Linear Regression and Random Forest) for each heuristic of the prediction of gaps (fourth column), end time jobs (fifth column) and makespan (sixth column) on different datsets (Uniform, Gaussian and Beta).

			RMSE		
Dist.	Alg.	Heur.	Gap	End Time	Makespan
U	LR	SPT	0.58	0.41	0.92
U	LR	LPT	0.74	0.84	1.60
U	RF	SPT	0.43	0.30	0.84
U	RF	LPT	0.62	0.71	1.01
G	LR	SPT	0.60	0.48	1.20
G	LR	LPT	0.71	0.78	1.49
G	RF	SPT	0.50	0.43	1.20
G	RF	LPT	0.61	0.69	1.06
B	LR	SPT	0.59	0.29	0.30
B	LR	LPT	0.70	0.48	0.42
B	RF	SPT	0.48	0.25	0.29
B	RF	LPT	0.60	0.44	0.33

4.1. Discussion

The results on the problem of predicting the gaps are generally positive, especially the ones obtained with the Random Forest algorithm (Table 1 and Figure 2). Better results are obtained in the prediction of the gaps generated with SPT than with LPT. This could either mean that the latter is a more difficult problem or that the features are more suitable for the former problem.

Some interesting observations can be made based on Figure 2 and further analysis of the results (not shown due to lack of space). First, there are a lot of examples (gaps) with a true value of 0. On the other hand, although the maximum value of the gaps is approximately 500, the predictions are generally below 300. We observe that the largest errors occur at the extremes of the range of real values. These observations indicate that the regression methods are not dealing with the distribution of target values appropriately.

Additionally, we present the results on the intermediate steps. For the Beta dataset, the error of predicting the gaps is higher than the error of predicting the end time of jobs for both heuristics (Table 1 and Figure 4). Additionally, the results obtained for SPT are clearly better than for LPT. Figure 3, which plots the distribution of the errors of predicting gaps for the two heuristics, provides a possible explanation for this. The distribution for SPT is more symmetric than for LPT. So, when adding the gaps, the errors of heuristic SPT may be cancelling themselves out because of the symmetry around 0. However, further analysis of results is necessary to confirm this.

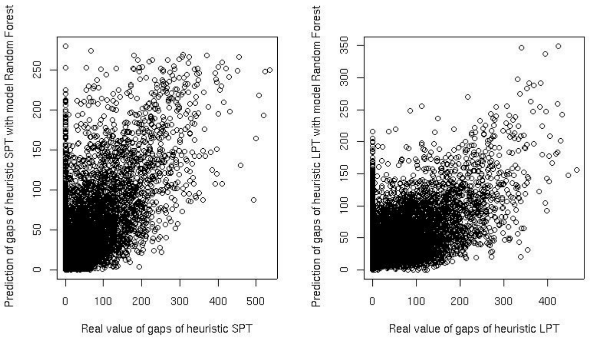

Figure 2. Graphic of value prediction using Random Forest model (x-axis) and real value of gaps duration for SPT(left) and LPT(right) of Beta dataset. Similar plots were obtained for the Uniform and Gaussian datasets.

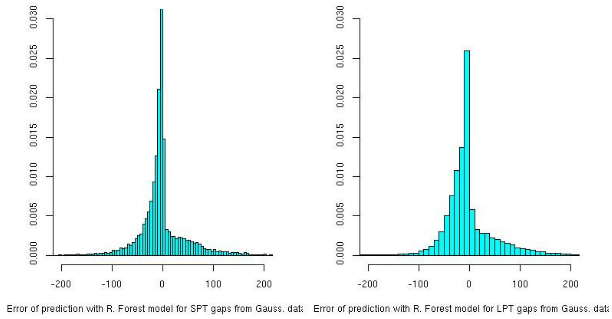

Figure 3. Histograms of the errors prediction with R. Forest model for SPT gaps (left) and LPT (right) from Gauss. dataset

We also analyse the results on the the prediction of the makespan of each heuristic (Table 1 and Figure 5). In comparison to the error of predicting the end time of jobs, in the Uniform and Gaussian dataset, it increases while on the Beta dataset they are both very similar.

These results indicate that it is possible to predict the gaps generated by these two heuristics, which is the goal of this paper. However], we also evaluated the accuracy of selecting the best heuristic based on the estimated makespans. The results in Table 2 show that this is also possible. However, the difference to the default accuracy (i.e., using a baseline method that always selects the heuristic that wins in the largest number of instances in the training set) is small. The table also shows that in instances with a larger difference of performance between the heuristics, the advantage of the prediction is more clear. These results indicate

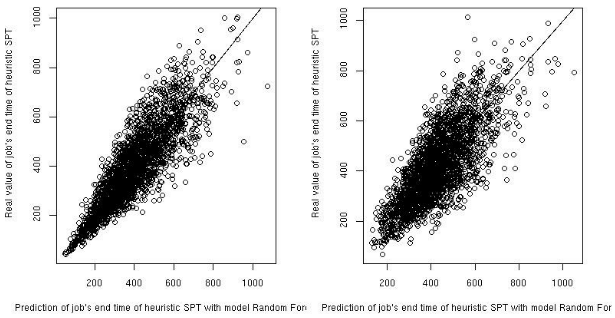

Figure 4. Graphic of value prediction using Random Forest model (x-axis) and real value of end time of the job's for SPT(left) and LPT(right) of Beta dataset. Similar results were obtained on the Uniform and Gaussian datasets.

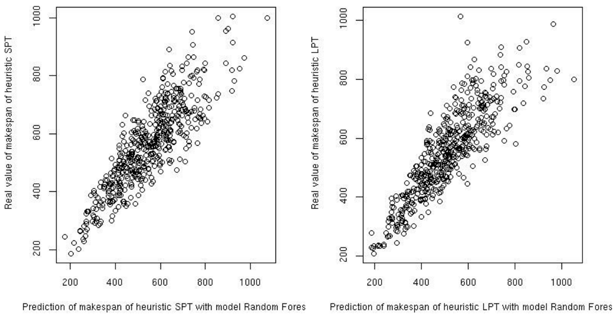

Figure 5. Graphic of value prediction using Random Forest model (x-axis) and real value of makespan of the instances for SPT(left) and LPT(right) of Beta dataset. Similar results were obtained on the uniform and gaussian datasets.

that we must improve the predictions of the gaps before being able to accurately predict the makespan and which method will obtain the best result.

One important goal of this work is to show that it is possible to use the approach proposed here to obtain information about the behavior of heuristics for the JSS problem. To illustrate this, we analyze the coefficients of the linear model obtained for the LPT heuristic on the Gaussian dataset. The coefficients with the largest effect are both negative, meaning that higher values of the corresponding variables are associated with a lower value of the gap. The two features represent the average of duration of the operations with higher duration than the operation that precedes the gap and with an ICS distance between 0 and 1.5 and -1.5 and 0, respectively. This means that the duration of the gap is inversely proportional to the duration of the operations which are very near in the ICS to the operation that precedes the gap (i.e., ICS distance between -1.5 and 1.5) and that have a higher priority for the LPT heuristic (i.e., they have a longer duration). This is an interesting result for two reasons. Firstly, it indicates that the ICS provides useful information about the behavior of the LPT heuristics, at least in small instances. It remains to be investigated whether the same is true for larger instances. Secondly, one hypothesis for this effect is that, under these conditions, there is a high probability that the operation that precedes the gap will be in conflict with operations that have higher priority according to LPT. Therefore, the gap *before* the operation is probably longer, which possibly decreases the probability that there will be a conflict *after* that operation. In other words, the gap which is being predicted is expected to be shorter. This indicates one possible problem with our approach. The features describe the operation *before* the gap, while, the operation *after* it is expected to have the largest effect on its size. Further work is reuquired to test this hypothesis.

Table 2. Error results obtained with different models (Linear Regression, Random Forest and Baseline) for each dataset type of the prediction of best heuristic for all instances and for selected instances, i.e., the ones for which there is a clear decision, identified by a difference in the predicted makespan higher than 20%.

Dist.	Alg.	All	Selected
U	LR	0.38	0.23
U	RF	0.33	0.15
U	Bl	0.43	0.33
G	LR	0.39	0.25
G	RF	0.39	0.24
G	Bl	0.42	0.31
B	LR	0.39	0.37
B	RF	0.34	0.17
B	Bl	0.46	0.37

The highest positive coefficient is the one of the feature representing the third quartile of the duration of the operations with higher duration (i.e., with a higher priority for the LPT) and with an ICS distance smaller than -3 (i.e., farther from the operation being considered). This result may be related to the previous one. It may indicate that, if there is less probability of a conflict with the operation before a gap (because the operations with higher priority have been scheduled a lot earlier), then the probability that there is a conflict in the following operation increases and so does the gap.

5. Conclusion

In this work, we addressed the problem of learning models that are able to predict behavior of different heuristics for the Job-Shop Scheduling problem. We propose a methodology that is novel and can be applied to any heuristic that solves conflicts individually. Our goal is to determine the properties of the instance that determine the performance of the heuristic. Our approach divides the problem into two sub-problems: 1) for every pair of consecutive operations, predict the size of the corresponding gap in the schedules generated by the heuristic and 2) predict the makespan of the schedule based on the predictions made in sub-problem 1.

In our experiments, positive results were obtained in the two sub-problems. However, the results on the second problem are not entirely satisfactory. Our plan is to improve the set of features used in order to obtain more accurate predictions in the first sub-problem. We expect that better base-level predictions will enable better results on the prediction of the makespan of the schedules generated by the heuristics.

As future work, we plan to explore the generality of the method, by testing it with another common heuristic for the JSS problem, MWR, and on heuristic methods that generate non-delayed schedules. It's important to analyze and understand the models for generate better features. Finally, we will also extend this work to predict the behavior of meta-heuristics. In this case, these results are particularly interesting because, as these heuristics are more computationally complex, the problem of selecting beforehand which method to apply is quite relevant.

Acknowledgments

This work was partially supported by project *Hybrid optimization and simulation methods for the Job-Shop problem and its sub-problems*, funded by Fundação para a Ciência e Tecnologia (POCI/EGE/61823/2004).

References

Abreu, P., & Soares, C. (2007). Mapping charateristics of instances to evolutionary algorithm operators: an empirical study on the basic job-shop scheduling problem. *Proceedings of the CAEPIA 2007 Workshop on Planning, Scheduling and Constraint Satisfaction* (pp. 80–92).

Baker, K. R. (1974). *Introduction to sequencing and scheduling*. New York: Wiley.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of np-completeness*. San Francisco, California: W. H. Freeman.

Giffler, B., & Thompson, G. (1960). Algorithms for solving production scheduling problems. *Operations Research*, *8*, 487–503.

Jain, A. S., & Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, *113*, 390–434.

Kononenko, I., & Kukar, M. (2007). *Introduction to machine learning and data mining: Introduction to principles and algorithms*. Horwood Publishing.

Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Computational complexity of discrete pptimisation problems. *Annals of Discrete Mathematics*, *4*, 121–140.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*, 278–285.

Watson, J.-P., Barbulescu, L., Howe, A. E., & Whitley, D. (1999). Algorithm performance and problem structure for flow-shop scheduling. *AAAI/IAAI* (pp. 688–695).

Meta-learning with kernels and similarity functions for planning of data mining workflows

Alexandros Kalousis KALOUSIS@CUI.UNIGE.CH
Abraham Bernstein BERNSTEIN@IFI.UZH.CH
Melanie Hilario HILARIO@CUI.UNIGE.CH

Keywords: planning, meta-learning, kernels, similarities, relational learning

Abstract

We propose an intelligent data mining (DM) assistant that will combine planning and meta-learning to provide support to users of a virtual DM laboratory. A knowledge-driven planner will rely on a data mining ontology to plan the knowledge discovery workflow and determine the set of valid operators for each step of this workflow. A probabilistic meta-learner will select the most appropriate operators by using relational similarity measures and kernel functions over records of past sessions meta-data stored in a DM experiments repository.

1. Introduction

We propose an architecture that combines a planning-based and a meta-learning approach in providing data mining support to end users. By adding AI-planning to meta-learning, we can ensure support for the complete knowledge discovery process. Contrary to previous efforts where the dominant focus was on either learning (Statlog, Metal) or preprocessing (MiningMart), our data mining assistant will propose workflows that start with the raw data, select and sequence the different preprocessing operations, select a suitable learning algorithm and output trained models. On the other hand, by adding meta-learning to planning-based data mining (DM) support, it will make the planner adaptive to changes in the data and capable of improving its advice over time; this improvement will apply to the planner's decision-making at any node of the knowledge discovery workflow.

Meta-learning will be based on multiple and diverse types of meta-data. Statlog and Metal meta-learners relied mainly on quantitative (e.g., statistical, information-theoretic) characteristics of data to select appropriate learning algorithms (Michie et al., 1994; Metal, 2002). MiningMart described datasets in terms of domain concepts but did not use these to metalearn (Morik & Scholz, 2004). The proposed system will meta-mine both quantitative and qualitative, domain ontology based metadescriptions of the application dataset. In addition, the meta-learner's ken will, for the first time, go beyond the dataset to take into account a significantly extended learning context the application task, performance criteria, workflow quality indicators, and the user's profile as defined by quantitative results and qualitative feedback from his past historical record of data mining experiments.

Generalizing from these heterogeneous factors requires defining similarity measures and data mining operators over complex structures. We will explore elaborate task descriptors such as operator trees (Mierswa et al., 2006) or multirelational experiment descriptors that integrate information concerning datasets, algorithms, and evaluation strategies, (Kalousis & Hilario, 2003; Hilario & Kalousis, 2001).

We propose a novel meta-learning technique which blends probabilistic reasoning and kernel-based learning from complex structures. We will exploit a framework that we have recently developed for kernel-based learning over complex structures using the language of relational algebra (Woznica et al., 2007; Woznica et al., 2005). To meta-learn from the diverse factors described above, we will weave state transition probabilities into kernel-based learning over relational schemas and devise methods for adjusting these probabilities to improve the data mining assistant's choices as it gains experience.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. An Intelligent Assistant for Data Mining

The intelligent data mining assistant will be at the helm of a virtual DM laboratory designed for a community of users with common data-analytical needs in a specific application domain. Essential components of this e-lab will be a DM ontology and a DM experiments repository. The ontology will provide a formal specification of the knowledge discovery process – its different phases, the set of operators that can be legitimately applied at each phase, and so forth. The repository will be the e-lab's long-term memory; detailed records of all experiments performed in the e-lab will be stored in the repository to allow for replication and comparative meta-analysis of data mining experiments.

The DM assistant will take in user specifications of the knowledge discovery task and available data, plan a methodologically correct learning process, and suggest ranked workflows that the user can enact to achieve the pre-specified objectives. To plan the workflow and determine the operator or algorithm to apply for a given data mining step, the assistant will harness prior knowledge stored in the DM ontology. Metadata stored in the DM experiments repository will be leveraged to improve the data mining process itself, for instance by incrementally refining the DM planner's search in the design space of candidate DM operators (and workflows). The kernel-based, probabilistic meta-learner will dynamically adjust transition probabilities between DM operators, conditioned on the current application task and data, user-specified performance criteria, quality scores of workflows applied in the past to similar tasks and data, and the user's profile (based on quantified results from, and qualitative feedback on, her past DM experiments). The proposed meta-learning method will be evaluated against the baseline of a case-based DM planner, which retrieves and adapts workflows from the most similar past experiments. By comparing the DM planner's evolution over time based on these two approaches, we hope to gain insights into the patterns that govern the efficacy of data mining workflows, operators and parameters.

2.1. The Knowledge Driven Planner

A DM ontology and repository will ease the task of constructing a complicated KD process by simplifying scientists' access to the plethora of data mining concepts, algorithms, data sources, and past experiments.

We will devise a tool that helps data miners (and data mining scientists) to navigate the space of KD processes systematically, and more effectively. In particular, we will develop an intelligent discovery assistant (IDA) that helps a data miner with the exploration of the space of valid[1] DM processes (Bernstein & Provost, 2005). The discovery assistant's intelligence comes mainly from its awareness of the full knowledge discovery context and its capacity to learn incrementally from experience. The KD context is available to the IDA in the form of the user's task specification and domain-ontology based semantic annotations on the dataset. In addition, the IDA can extract quantitative characteristics of the dataset such as the number of explanatory variables or the percentage of missing values. The IDA uses this contextual information, together with knowledge from the DM ontology and knowledge base (e.g., applicability conditions of DM operators), to search for and enumerate the valid and effective DM processes. It does this by (i) retrieving and adapting them from the DM experiments repository using case-based reasoning approaches, or (ii) using AI planning type approaches to construct new valid data mining processes.

Once the IDA has listed a variety of alternatives it also assists the user in choosing workflows to execute, for example, by ranking the workflows (heuristically) according to what is important to the user. In addition, the IDA will also allow for some open-ended, statistical/exploratory data analysis, as has been addressed by Amant and Cohen (1998). In such explorations, the IDA does not necessarily provide the user with finished KD-workflows, but provides guidance at each step in the exploration of a KD-process - a type of support that is suitable in data mining endeavors that are exploratory and/or where the case-based/planning based IDA does not provide satisfactory KD-workflows. In this exploratory mode, scientists (or data miners) would first assemble underlying data-sources after which the IDA would try to provide advise on what possible next steps could be. As soon as scientists would choose one of these steps the IDA would execute it in the background and try to advise on next steps or suggest backtracking to the previous decisions if some newly arisen information would warrant this.

To allow the planner-based IDA to improve with experience, we introduce a set of probabilistic parameters that will be automatically adjusted by meta-

[1] A *valid* DM workflow violates no fundamental constraints of its constituent techniques. An automated system can take advantage of an explicit ontology of data-mining techniques, which defines the various techniques and their properties.

mining the DM experiments repository. The DM-workflow planner is essentially a breadth-first search algorithm that starts from an initial state and tries to reach a final stage by sequencing data mining operators. At each state the search algorithm will add all DM operators that can be legitimately applied. Prior knowledge about operator application constraints is obtained from the DM ontology and modeled in a state transition table $\mathbf{T}$ with dimensionality $K \times K$ (K is the number of DM operators). The T_{ij} element of this table is defined as $T_{ij} = P(O_j|O_i, \mathbf{D}, \mathbf{KDT})$. In words, it denotes the transition probability from state (DM operator) i, to state (DM operator) j, given the description $\mathbf{D}$ of the data and the description of the knowledge discovery task, $\mathbf{KDT}$. These probabilities sum to one over a given row i, $\sum_j T_{ij}$. In the simple breadth-first search all valid transitions are equiprobable (will all be expanded and explored), since there are no preferred sequences of operators.

The planner should establish the sequence of data mining operators $WF = [S_1, S_2, ..., S_N]$ with maximal joint probability distribution given the data description D, and the knowledge discovery task description, that is: $WF = argmax_{WF} P(S_1, S_2, ..., S_N|\mathbf{D}, \mathbf{KDT})$. Under the assumption that the transition to the next stage depends only on the current stage, the data description, $\mathbf{D}$, and the knowledge discovery task, $\mathbf{KDT}$, the joint probability distribution $P(S_1, S_2, ..., S_N|\mathbf{D}, \mathbf{KDT})$ factorizes as: $P(S_1|\mathbf{D}, \mathbf{KDT}) \prod_{i=2}^{N} T_{S_{(i-1)}S_i}$ The initial stage S_1 is governed by a probability distribution defined over the different data mining operators, given the data $\mathbf{D}$ and the knowledge discovery task $\mathbf{KDT}$, i.e. $S_1 \sim P(\mathbf{O}|\mathbf{D}, \mathbf{KDT}) = [P(O_i|\mathbf{D}, \mathbf{KDT})|i = 1 \dots N]$. It is straightforward to adapt the search algorithm to output together with the DM-workflows their joint probabilities. Assuming $\mathbf{T}$ is in its original state, i.e. ignoring all past experiments in the DM experiments repository, all DM workflows of equal length will also have equal joint probability, as the breadth search algorithm examines all states without distinction. Note here that longer workflows will have lower joint probability than shorter since their joint probability is a product of a larger number of terms, one issue that arises here is whether this should be factored out by an appropriate normalization, nevertheless intuitevely one would prefer shorter and simpler workflows over more complex.

We have expressed the way in which the search algorithm moves around the space of states, i.e. data mining operators, in terms of state transition probabilities, i.e the set of parameters $\mathbf{T}$, $P(\mathbf{O}|D, KDT)$. It is this set of parameters that will be the target of the incremental meta-learning.

2.2. Meta-Learning

The general problem that IDA tries to solve can be formulated as follows: given a user, U, of the platform facing a knowledge discovery problem, $\mathbf{A}$, and a description of $\mathbf{A} = (\mathbf{D}, \mathbf{KDT})$, where, $\mathbf{D}$ corresponds to a description of the data, both in terms of their semantics, as these are established after their annotation with respect to the domain ontology, and their quantitative characteristics, $\mathbf{KDT}$ is a description of the Knowledge Discovery Task that the user is trying to accomplish, establish a data mining workflow $\mathbf{WF}$ which will address the knowledge discovery task and will optimize some performance criteria, $\mathbf{PC}$, specific to the user. The description of $\mathbf{KDT}$ can be as high level as simply stating the learning task to be performed, e.g. classification, or more specific such as stating that the goal is classification using a reduced set of features. Its description will be given in the form of a workflow although at a high level. At the beginning IDA will rely solely on the data mining ontology and the planner to propose and rank a number of alternative WFs; essential to the planning process is the set of parameters, $\mathbf{T}$, $P(\mathbf{O}|D, \mathbf{KDT})$. The initial state of these parameters will be determined by the data mining experts. To allow the planner to recommend the most appropriate workflow(s), we propose an infrastructure to adapt these parameters to the requirements of a given knowledge discovery problem as these can be gathered from $(\mathbf{A}, U, \mathbf{PC})$. The adaptation process will take account of previous data mining experiments and performance results, as well as other factors such as users' feedback, context and reputation.

More precisely, the system is confronted with a number of data mining experiments performed by various users, which are eventually stored in the DMER. Each data mining experiment, $\mathbf{DME_k}$, is a complex structure described by a number of components that will eventually resemble to something like: $\mathbf{DME_k} = (U_k, \mathbf{WF_k}, \mathbf{D_k}, \mathbf{KDT_k}, \mathbf{PC_k}, \mathbf{UF_k})$. U_k is the identifier of the user that performed the given experiment, the remaining variables have a complex structure. $\mathbf{WF_k}$ denotes the workflow that was applied on the given $\mathbf{DME_k}$, and is actually a sequence of data mining operators; $\mathbf{D_k}$ is the description of the data analysed in the experiment$\mathbf{DME_k}$; $\mathbf{KDT_K}$ is the description of the knowledge discovery task that is to be performed; $\mathbf{PC_k}$ is a vector containing different performance measurements obtained by applying workflow WF_k to dataset $\mathbf{D_k}$, with respect to a number of performance criteria; and $\mathbf{UF_k}$ is some qualitative user feeback along a number of different dimensions, such as understandability, ease of use, complexity etc.

Learning the planner's parameters for a new Knowledge Discovery Problem The meta-learning module will establish functions $f_{\mathbf{T}}(\mathbf{A})$ and $f_{\mathbf{O}}(\mathbf{A})$ that, given the description, $\mathbf{A} = (\mathbf{D}, \mathbf{KDT})$, of a new, potentially unseen, knowledge discovery problem, will estimate $\mathbf{T}$ and $P(\mathbf{O}|\mathbf{A})$, respectively. These estimates will then be used by the planner to provide a ranked list of workflows for $\mathbf{A}$. The main learning paradigm that we will use is that of kernel-based estimation. Let $\mathbf{X_k} = (\mathbf{D_k}, \mathbf{KDT_k})$ denote the description of the knowledge discovery problem associated with data mining experiment $\mathbf{DME_k}$, then:

$$f_{\mathbf{T}}(\mathbf{A}) = \frac{\sum_{\mathbf{X_k}} \overline{\mathbf{T_k}} K_{KDP}(\mathbf{A}, \mathbf{X_k})}{\sum_{\mathbf{X_k}} K_{KDP}(\mathbf{A}, \mathbf{X_k})}, \tag{1}$$

$$f_{\mathbf{O}}(\mathbf{A}) = \frac{\sum_{\mathbf{X_k}} \overline{\mathbf{P(O|X_k)}} K_{KDP}(\mathbf{A}, \mathbf{X_k})}{\sum_{\mathbf{X_k}} K_{KDP}(\mathbf{A}, \mathbf{X_k})}$$

where the summations are taken over all $\mathbf{X_k} \in DMER$. $K_{KDP}(\mathbf{A}, \mathbf{X_k})$ is a kernel[2] function that provides a measure of similarity of $\mathbf{A}$ and $\mathbf{X_k}$; $\overline{\mathbf{T_k}}$ and $\overline{P(\mathbf{O}|\mathbf{X_k})}$ are estimations of $\mathbf{T}$ and $P(\mathbf{O})$ derived from $\mathbf{DME_k}$. The workflow, $\mathbf{WF_k}$, of a data mining experiment, $\mathbf{DME_k}$, can give rise to an estimation, $\overline{\mathbf{T_k}}$, of $\mathbf{T}$ simply by counting the times that a transition happens from one operator, O_i, to another operator, O_j, within that workflow and normalizing it by the total number of transitions. Similarly it can provide us with an estimation of $\overline{P(\mathbf{O}|\mathbf{X_k})}$. In both estimates we can imagine adding a Laplace correction so that the probability of the transitions that do not appear is greater than zero.

Kernels on Descriptions of Knowledge Discovery Problems The estimates given in equations 1 can be readily used by the planner to construct workflows which are tailored to $\mathbf{A}$. We will design kernel functions which are appropriate for this type of problem exploiting similarity measures defined over datasets in the context of meta-learning, (Kalousis & Hilario, 2003), but also more general kernel functions for complex objects, (Woznica et al., 2007; Woznica et al., 2005) Since the description of a knowledge discovery problem consists of two quite different parts, the data description and the knowledge discovery task description, we envisage that the $K_{KDP}(\mathbf{A}, \mathbf{X})$ kernel will be the composition of two very different kernels, one, K_D, defined on the data part of the description and another, K_{WF}, defined on the knowledge discovery task description part. The first kernel will have to account for similarities defined not only with respect to quantifiable characteristics of the datasets, but also with respect to their annotations within the domain ontology. The second kernel will be defined over the language used to described knowledge discovery tasks, potentially over different abstraction levels, and will exploit similarities of operators and concepts derived from the data mining ontology. Note here that the definition and availability of these kernel—similarity—functions will also serve the needs of the case-base and will result in similarity measures for the retrieval of similar knowledge discovery problems, datasets, and workflows—knowledge discovery tasks. We will design, test, and evaluate different ways of defining the K_{KDP} kernel, exploring and/or even learning the importance of its constituents, (Woznica et al., 2007). We will also explore different estimations of $\overline{\mathbf{T_k}}$ and $\overline{P(\mathbf{O}|\mathbf{X_k})}$ from a given workflow description $\mathbf{WF_k}$.

Accounting for Qualitative and Quantitative Performance Indicators of WFs The estimations derived from the equations of 1 are based only on the similarity of the description of the current knowledge discovery problem $\mathbf{A}$ with the descriptions of the knowledge discovery problems $\mathbf{X_k}$, thus ignoring any quality indicator for the $\mathbf{WF_k}$ workflow associated with $\mathbf{X_k}$. The quality indicators come into two flavors: quantitive performance measures, contained in $\mathbf{PC_k}$, that are estimated from the actual application of the $\mathbf{WF_k}$ workflow on the data, and qualitative indicators, contained in $\mathbf{UF_k}$, that are given by the user U_k, concerning non-easily quantifiable dimensions such as understandability and/or simplicity of the final models produced by the workflow, ease of use of the workflow etc. Yet a third, indirect, quality indicator of a workflow $\mathbf{WF_k}$ can come from the "quality" of the user U_k associated with the workflow. The quality of a user U_k will be given by a function $Q(U_k)$ that will account for various factors, such as how often workflows designed by this user have been adopted by other users, how the workflows of this user have been qualified by other users, how this user has been qualified by other users, etc. By accounting for such quality indicators of a $\mathbf{WF_k}$ workflow we can accordingly favor or penalise the estimations $\overline{\mathbf{T_k}}$ and $\overline{P(\mathbf{O}|\mathbf{X_k})}$ derived from $\mathbf{WF_k}$.

Moreover we should account for the fact that different users might have different preferences concerning the desired quantitative and qualitative performance indicators of the workflows, e.g. trading accuracy for

[2] A kernel function, $k(x, y)$, provides the similarity of the images of x and y in some feature space without having to compute explicitly the mapping.

understandability. In order to address such differences we will design user dependent parameterizable functions $f_u(\mathbf{PC_k}, \mathbf{UF_k})$ of the quality indicators. These functions will weight heavily the $\overline{\mathbf{T_k}}$ and $\overline{P(\mathbf{O}|\mathbf{X_k})}$ estimations derived from workflows that exhibit the desired performance while they will reduce towards zero the weights of estimations derived from workflows with poor performance. Incorporating these functions into equations of 1 results in estimates of the form:

$$f_{\mathbf{T}}(\mathbf{A}) = \frac{\sum_{\mathbf{X_k}} \overline{\mathbf{T_k}} K_{KDT}(\mathbf{A}, \mathbf{X_k}) f_u(\mathbf{PC_k}, \mathbf{UF_k}) Q(U_k)}{\sum_{\mathbf{X_k}} K_{KDT}(\mathbf{A}, \mathbf{X_k}) f_u(\mathbf{PC_k}, \mathbf{UF_k}) Q(U_k)} \tag{2}$$

$$f_{\mathbf{O}}(\mathbf{A}) = \frac{\sum_{\mathbf{X_k}} \overline{\mathbf{P}(\mathbf{O}|\mathbf{X_k})} K_{KDT}(\mathbf{A}, \mathbf{X_k}) f_u(\mathbf{PC_k}, \mathbf{UF_k}) Q(U_k)}{\sum_{\mathbf{X_k}} K_{KDT}(\mathbf{A}, \mathbf{X_k}) f_u(\mathbf{PC_k}, \mathbf{UF_k}) Q(U_k)}$$

We will thus design, test and evaluate, performance aware estimates of the parameters of the planner, according to the equations of 2 by incorporating user dependent functions of qualitative and quantitative performance indicators of the workflows as well as user quality indicators through the $Q(U_k)$ function. The latter will draw heavily on the definition of authority indexes described later.

User's Profile—Context The profile of a user consists of the information stored about the user within the system. This information consists of all the previous knowledge discovery projects that he/she has undertaken, the data mining experiments that he performed within each knowledge discovery project, the datasets associated with these experiments, the descriptions of these datasets, the workflows that he/she has chosen for final deployment or publication on the platform, the feedback that he/she has provided on previous suggestions of the system, the workflows he/she has designed from scratch without relying on the system's support, his/her level of authority, as this is determined by the frequency of use by other users of the workflows he/she has published. We will define precisely the different information sources that collectively constitute a user's profile. An important part of this task will be the definition of authority indexes for the users of the system. We will construct kernel functions, $K_U(U_k, U_l)$, to measure the similarities of the profiles—contexts— of any pair of users, U_k, U_l. Since the profile of a user consists of datasets, knowledge discovery task descriptions, and more, the K_U kernel will be actually a set kernel based on agreegations of K_{KDP} kernels on the different problems that the user has encountered, potentially including other kernels defined on other aspects of a user's profile.

Incorporating User's Feedback and Context So far the estimations of the parameters given by $f_{\mathbf{T}}(\mathbf{A})$ and $f_{\mathbf{O}}(\mathbf{A})$ were adapted to the descriptions of the knowledge discovery problem that should be solved, accounting for the quality of the previous solutions, but they do not incorporate any existing feedback from the user that is performing the current data mining experiments on previous analysis episodes and workflows within there, nor any information about his/her profile. In order to do that we will incorporate the $K_U(U_k, U_l)$ kernel in the computation of $f_{\mathbf{T}}(\mathbf{A})$. Like that the qualitative and quantitative indicators given by the user U, who is performing the actual experiment, in his/her past interactions with the system, will be given maximum weight since $K_U(U, U)$ atains the maximum possible similarity value. Moreover the incorporation of $K_U(U, U_k)$ will assign greater importance to users that exist in similar contexts as the U, thus modeling the assumption that users in similar context will probably find interesting similar tools. The new estimations will be functions of both the description of the knowledge discovery application problem, $\mathbf{A}$, and the user, U, who is faced with $\mathbf{A}$.

$$f_{\mathbf{T}}(\mathbf{A}, U) = \frac{\sum_k \overline{\mathbf{T_k}} K_{KDT}(\mathbf{A}, \mathbf{X_k}) f_u(\mathbf{PC_k}) f_u(\mathbf{UF_k}) Q(U_k) K(U, U_k)}{\sum_k K_{KDT}(\mathbf{A}, \mathbf{X_k}) f_u(\mathbf{PC_k}) f_u(\mathbf{UF_k}) Q(U_k) K_U(U, U_k)} \tag{3}$$

$$f_{\mathbf{O}}(\mathbf{A}, U) = \frac{\sum_k \overline{\mathbf{P}(\mathbf{O}|\mathbf{X_k})} K_{KDT}(\mathbf{A}, \mathbf{X_k}) f_u(\mathbf{PC_k}) f_u(\mathbf{UF_k}) Q(U_k) K(U, U_k)}{\sum_k K_{KDT}(\mathbf{A}, \mathbf{X_k}) f_u(\mathbf{PC_k}) f_u(\mathbf{UF_k}) Q(U_k) K_U(U, U_k)}$$

We will define, test, and evaluate, the final form of the adaptive estimations of the parameters of the planner which will incorporate the user's feedback on previous analysis episodes and suggestions of the system, as well as information about the user's context similarity with that of other users. In the latter the idea is that users that exist in similar contexts will have similar data analysis needs.

3. Evaluation

We will systematically evaluate the different strategies for estimating the parameters of the DM-workflow planner. The basic evaluation strategy will be to examine how well the suggestions of the planner, under the different estimation strategies, correlate with the users' actual feedback. Standard hold-out or resampling-based strategies will be used to estimate this correlation. The key idea will be to use a part of the available data to build the estimates for unseen cases and compute the correlation with the user feedback. Different levels of evaluation will be of interest, namely, evaluating performance on completely new users for which nothing is known, and evaluating performance for users that have a recorded history.

4. Discussion

In this paper we propose a system that will combine planning and meta-learning to provide support to users of a virtual laboratory. Standard planning approaches return a number of different solutions, typically unranked. Our planner will rank these solutions according not only to their probabilities among different users and different user communities, as these are depicted in the state transition matrix, but also with respect to a number of qualitative and quantitative performance indicators on past problems. Equally important we account for the different degrees of relevance that these performance indicators might have for different users, or even for the same user in different contexts, by incorporating as a part of the establishement of the final ranking of plans parameterizable, according to user preferences, functions of these quality indicators.

A crucial factor for the success of the system, especially if one considers the enormous size of hypothesis space of the metalearning problem, is the construction of a large repository of Data Mining Experiments. In order to address this issue we plan to exploit ideas from social networking coupled with e-science platforms. One such platform is MyExperiment, (Goble & De Roure, 2007), which is an e-science social network that supports the exchange of complex workflows that address bioinformatics problems. Exploiting the idea of social networks for the construction of a network of Data Mining Scientists provides a very promising way to address the problem of data collection for metalearning. Such social networks already exist in the form of forums build around specific data analysis tools such as Weka (Witten & Frank, 2005), RapidMiner (Mierswa et al., 2006). Moving to the next stage where participants will exhange not only comments and suggestions but in fact complete data mining workflows, such as Weka or RapidMiner workflows, that can be readily applied is not such a great leap. We believe that the benefits for the data analysis community would be great and analysts will have every reason to participate to such a community by contributing content benefiting from the collective intelligence.

References

Amant, S., & Cohen, P. (1998). Intelligent support for exploratory data analysis. *Journal of Computational and Graphical Statistics*, *7*, 545–558.

Bernstein, A., & Provost, F. andHill, S. (2005). Towards intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering*, *17*.

Goble, C., & De Roure, D. (2007). myexperiment: social networking for workflow-using e-scientists. *In Proceedings of the 2nd workshop on Workflows in support of large-scale science.*

Hilario, M., & Kalousis, A. (2001). Fusion of meta-knowledge and meta-data for case-based model selection. *In Proceedings of the Fifth European Conference on Principles and Practice of Knowledge Discovery in Databases* (pp. 180–191).

Kalousis, A., & Hilario, M. (2003). Representational issues in meta-learning. *Proc. 20th International Conference on Machine Learning (ICML-2003)* (pp. 313–320). Morgan Kaufmann.

Metal (2002). Metal related bibliography. http://www.ofai.at/research/impml/metal/metal-publications.html.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (Eds.). (1994). *Machine learning, neural and statistical classification.* Ellis-Horwood.

Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., & Euler, T. (2006). YALE: Rapid prototyping for complex data mining tasks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006).*

Morik, K., & Scholz, M. (2004). *Intelligent technologies for information analysis*, chapter The MiningMart Approach to Knowledge Discovery in Databases. Springer.

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques.* Morgan Kaufmann.

Woznica, A., Kalousis, A., & Hilario, M. (2005). Kernels over relational algebra structures. *Proceedings of the Ninth Pasific Asia Conference on Knowledge Discovery, PAKDD.* Springer.

Woznica, A., Kalousis, A., & Hilario, M. (2007). Learning to combine distances for complex representations. *Proceedings of the 24th International Conference on Machine Learning.*

Planning to Learn with a Knowledge Discovery Ontology

Monika Žáková, Petr Křemen, Filip Železný {ZAKOVM1,KREMEP1,ZELEZNY}FEL.CVUT.CZ
Czech Technical University in Prague, Czech Republic

Nada Lavrač NADA.LAVRAC@IJS.SI
Jožef Stefan Institute, Ljubljana, Slovenia, and the University of Nova Gorica, Nova Gorica, Slovenia

Abstract

This paper addresses the problem of semi-automatic design of workflows for complex knowledge discovery tasks. Assembly of optimized knowledge discovery workflows requires awareness of and extensive knowledge about the principles and mutual relations between diverse data processing and mining algorithms. We aim at alleviating this burden by automatically proposing workflows for the given type of inputs and required outputs of the discovery process. The methodology adopted in this study is to define a formal conceptualization of knowledge types and data mining algorithms and design a planning algorithm, which extracts constraints from this conceptualization for the given user's input-output requirements. We demonstrate our approach in two use cases, one from scientific discovery in genomics and another from advanced engineering.

1. Introduction

Integration of heterogeneous data sources and inferring new knowledge from such combined information is one of the key challenges in present-day life science. Consider e.g. bioinformatics where for virtually any biological entity (a gene, for example), vast amounts of relevant background information are available from public web resources. This information comes in diverse formats and at diverse levels of abstraction. Continuing the genomic example, the publicly available data sources range from DNA sequence information, homology and interaction relations, gene-ontology annotations, information on the involvement in biological pathways, expression profiles in various situations etc. To merge only these exemplary sources of data, one already has to combine specialized algorithms for processing sequences, relational data, ontology information and graph data. It is thus no surprise that a principled fusion of such relevant data requires the interplay of diverse specialized algorithms resulting in highly intricate workflows. While the mutual relations of such algorithms and principles of their applicability may be mastered by computer scientists, their command cannot be expected from the end user, e.g. a life scientist.

The primary hypothesis investigated in our study is that such complex scientific workflows can be assembled automatically with the use of a knowledge discovery ontology and a planning algorithm accepting task descriptions automatically formed using the vocabulary of the ontology.

2. Related Work

Several previous works have explored planning in the context of workflows. Notably, within the Pegasus project (Deelman et al., 1990) a planner is used to construct a concrete workflow out of an abstract workflow. In our research we tackle a related yet different goal; given an ontology and a task description, we use a planner to construct a workflow, which in the terminology of (Deelman et al., 1990) would be called abstract. The paper (Klusch et al., 2005) is relevant to our work as it elaborates a procedure for converting OWL-S service annotations into action descriptions in the standard Planning Problem Description Language (PDDL). Constructing a PDDL problem description is also a technical ingredient of our methodology.

Unlike in (Klusch et al., 2005) we conduct workflow composition tasks in the specific domain on data mining, for which we devise a special ontology. A similar aim was followed by recent work of Brezany et

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

al. (Brezany et al., 2007). This work, however, is focused only on automatic formation of linear sequences of tasks: their ontology ensures that there is only one algorithm that can be inserted into a workflow prior to another algorithm. In our work we try to provide a more principled framework for the domain of data mining, aimed at enabling the construction of much more complex workflows with the main intended application in non-trivial scientific discovery tasks.

To the best of our knowledge, there is so far no previous work providing a principled and actionable ontology for data mining including relational data mining with complex background knowledge. There have been efforts to provide a systematic description of data and processes for the classical data mining tasks e.g. in systems MiningMart (Morik & Scholz, 2004), CAMLET (Suyama et al., 1998), CITRUS (Wirth et al., 1997) and NExT (Bernstein & Deanzer, 2007).

The MiningMart system (Morik & Scholz, 2004) focuses on propositional data mining from data stored in a relational database. It contains a meta-model for representing and structuring of information about data and algorithms, however this meta-model is expressed in XML, not in an ontology language. The system also does not provide means for automatic workflow creation.

The project CITRUS (Wirth et al., 1997) uses an object oriented schema is used to model relationships between the algorithms. The system focuses on guiding the user through mostly manual process of building of workflows by including information about properties and usability of the algorithm in the algorithm description. Planning is used only for proposing steps in process decomposition and refinement.

In the CAMLET system an ontology of algorithms (processes) and ontology of data structures are defined, however no ontology language is specified in (Suyama et al., 1998). The system relies on manually defined top-level control structure, which is then refined using genetic programming until a suitable structure producing the required results is found. The structure of algorithms ontology does not attempt to formalize the domain systematically, rather it is determined by the used top-level control structure.

The most systematic effort to construct a general knowledge discovery ontology is described in (Bernstein & Deanzer, 2007). The ontology used by the NExT system is built on OWL-S and provides a relatively detailed structure of the propositional data mining algorithms. It focuses on classical data mining processes, which contain three subsequent steps: pre-processing, model induction and post-processing, while our primary focus in on describing more complex relational data mining tasks. The workflows generated by the NExT system are linear, whereas for our tasks workflow is a directed acyclic graph.

The development of a unified theory (conceptualization) of data mining was recently identified as the first of ten most challenging problems for data mining research (Yang & Wu, 2006). While we do not claim completeness or universal applicability of the ontology developed in this work, in its design we did try to follow the state-of-the-art works attempting to establish such a unified theory. From Mannila's traditional definition of data mining (Mannila, 1995), we accepted the core concepts of a pattern and representation language. On the other hand, in categorizing knowledge and algorithm types, we followed on the recent comprehensive study by Džeroski (Dzeroski, 2007).

3. Knowledge Discovery Ontology

Our knowledge discovery ontology defines relationships among diverse ingredients of knowledge discovery scenarios, including both declarative (various knowledge representations) and algorithmic (both inductive and deductive algorithms). The primary purpose of the ontology is to make the workflow planner able to reason about which algorithms can be used to produce intermediary or final results required by a specified data mining task. Due to limited space, we constrain ourselves to describing only the basic aspects of our approach to designing the ontology, which essentially follows up on the recent attempts of establishing a conceptual framework for data mining (Dzeroski, 2007). Our proposal addresses two core concepts: *knowledge*, capturing the declarative elements in knowledge discovery, and *algorithms*, which serve to transform a piece of knowledge into another piece of knowledge. The basic top-level structure of the ontology is in Figure 1. Currently the ontology contains about 70 classes including descriptions of some propositional algorithms available in Weka data mining platform (Witten & Frank, 2005) and relational data mining algorithms described in (Žáková & Železný, 2007).

As an example of algorithm description we present the definition of the well known Apriori algorithm in the description logic notation (Baader et al., 2003) :

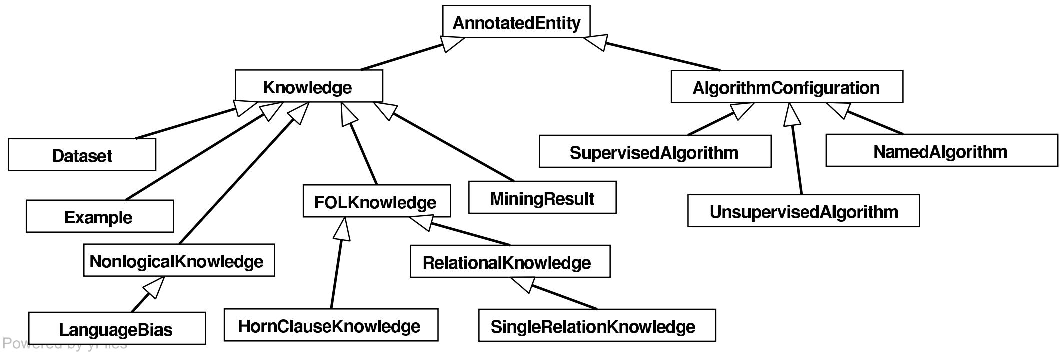

Figure 1. The basic top level structure of the knowledge discovery ontology.

$$\begin{array}{lll}
\texttt{Apriori} & \sqsubseteq & \texttt{NamedAlgorithm}\\
& \sqcap & \exists\,\texttt{output}\cdot(\texttt{MiningResult}\sqcap\\
& & \forall\,\texttt{contains}\cdot\texttt{AssociationRule})\\
& \sqcap & \exists\,\texttt{input}\cdot(\texttt{Dataset}\sqcap\\
& & \texttt{SingleRelationKnowledge}\sqcap\\
& & \exists\,\texttt{format}\cdot\{\texttt{ARFF},\texttt{CSV}\})\\
& \sqcap & \exists\,\texttt{minSupport}\cdot\textit{double}\\
& \sqcap & \exists\,\texttt{minConfidence}\cdot\textit{double}
\end{array}$$

The Apriori algorithm is defined as an algorithm that has two parameters `minSupport` and `minConfidence`, has a single relation dataset in the CSV or ARFF format as its input, and produces a result in the form of association rules.

Technically, the ontology is implemented in the description logic variant (OWL-DL) of the leading semantic web language OWL (Patel-Schneider et al., 2004). Our primary reasons for this choice were OWL's sufficient expressiveness, modularity, availability of ontology authoring tools and optimized reasoners and a well-established community support.

4. Workflow Construction

The task of automatic workflow construction consists of the following steps: converting the KD task into a planning problem, generating the plan using a third party planning algorithm, storing the generated abstract workflow in form of semantic annotation, instantiating the abstract workflow with specific configurations of the required algorithms and storing the generated workflow.

To maintain generality of our approach, we decided to encode the planning task into the standard language PPDL ('Planning Domain Definition Language') (Smith & Weld, 1999). We are using PDDL 2.0 with type hierarchy and domain axioms. Planning algorithms require two main inputs. The first one is a description of the domain specifying the available types of objects and actions. The second one is the problem description specifying initial state, goal state and the available objects. We have developed an algorithm for generating the domain description from the KD ontology. In order to formalize problem description and generate the problem description in PDDL in a similar way and for storing the created workflows in a knowledge-based representation, we have created a small ontology for workflows, which extends the knowledge discovery ontology.

As an example we present the definition of action in PDDL representing the Apriori algorithm described in Section 3.

```
(:action AprioriAlgorithm
 :parameters (
  ?v0 - Dataset_SingleRelationKnowledge
  ?v1 - CSV
  ?v2 - MiningResult_contains_Associa-
        tionRule )
 :precondition (and (available ?v0)
                    (format ?v0 ?v1))
 :effect (and (available ?v2)
              (format ?v2 ?v1)))
```

The information about the output of Apriori algorithm was expressed using a conjunction of the named ontological class *MiningResult*, a universal restriction on *contains* and an existential restriction on *format*. Therefore the effects of the action using Apriori algorithm are represented using the unary predicate *available* applied on a named class *MiningResult_contains_AssociationRule*, which is a subclass of

MiningResult, and a binary predicate *format*.

We have implemented a planning algorithm based on the Fast-Forward (FF) planning system (Hoffmann & Nebel, 2001) to generate abstract workflows automatically. The FF planning system uses a modified version of hill climbing algorithm called enforced hill climbing to perform forward state space search. The goal distances are estimated by relaxed GRAPHPLAN (Blum & Furst, 1997). The original planning problem is converted into a relaxed problem by ignoring delete lists of the operators.

Currently the planning algorithm outputs the first workflow with the smallest number of processing steps as the solution. In future work we are planning to include other heuristics such as the estimated runtimes of the workflows to provide the user with the possibility to view and select from a number of workflows with the highest ranking.

5. Use Cases

We have conducted experiments with workflow construction in two domains. The first domain is genomics, where we were interested in relational descriptive analysis of gene expression data. The second is concerned with learning from product design data. Here the examples are semantically annotated CAD documents.

Both these domains are highly knowledge-intensive. One of the main challenges is to efficiently extract relevant information from large amounts of data from different sources with a rich relational structure. The use of advanced knowledge engineering techniques is becoming popular not only in bioinformatics, but also in the engineering domain, and complex background knowledge thus nowadays characterizes both domains. As a result, traditional data mining techniques and tools are not straightforwardly applicable. Rather, complex knowledge discovery workflows are required in both the domains under inquiry.

An example of abstract workflow generated in the engineering domain is in Figure 2. There are four preprocessing steps, which can be performed simultaneously. In this case all the preprocessing steps focus on extracting knowledge from the semantic representation into a form, in which it could be used by relational data mining algorithm. *ModeDeclarationsExtractorOWLDL* extracts mode declarations from domain and range restrictions on properties defined in the CAD ontology. The sort theory containing a taxonomy of classes from the CAD ontology is extracted using *SortTheoryExtractorOWLDL*. The *OWLDLRelationalConverter* is used to convert descriptions of the individual annotations to Prolog and *RDF2PrologConverterCI* does the same for the identifiers of the annotations.

6. Conclusions and Future Work

We entered this study with the primary hypothesis that complex scientific and engineering knowledge discovery workflows, such as those we had to develop manually in previous studies (Trajkovski et al., 2008; Žaková et al., 2006), can be proposed semi-automatically. Semi-automatic workflow composition does require the user to know exactly what he/she possesses as the knowledge input and what kind of output he/she desires to achieve, but it does not require him/her to be aware of the numerous properties and mutual relationships of the wide range of relevant knowledge discovery algorithms.

For the purpose of workflow generation, we used two main ingredients. First, a formal conceptualization of knowledge types and algorithms was implemented through a *knowledge discovery ontology*, following up on state-of-the-art developments of a unified data mining theory. Second, a planning algorithm is employed that assembles workflows on the basis of planning task descriptions extracted from the knowledge discovery ontology and the given user's input-output task requirements. The workflows generated by our algorithm were complex, but reasonable in that there was no apparent way of simplifying them while maintaining the desired functionality. Therefore the workflows generated in two use cases (in science and engineering) can serve as a proof of concept for our approach.

Since the generated workflows are not linear, we could get significant runtime improvements from executing the workflows in the GRID environment. Therefore in future work we are planning to extend the ontology by descriptions of concrete computational resources e.g. by integration of our KD ontology into OWL-S. This will enable us to produce workflows optimized for execution in a given computing environment. We are also planning to evaluate the respective merits of planning via conversion into PDDL and building a planning algorithm directly over the DL representation.

Acknowledgements

This research was supported by the SEVENPRO, IST-027473 project in FP6 and by the 1ET101210513 project ML/BIO: Re- lational machine learning for analysis of biomedical data.

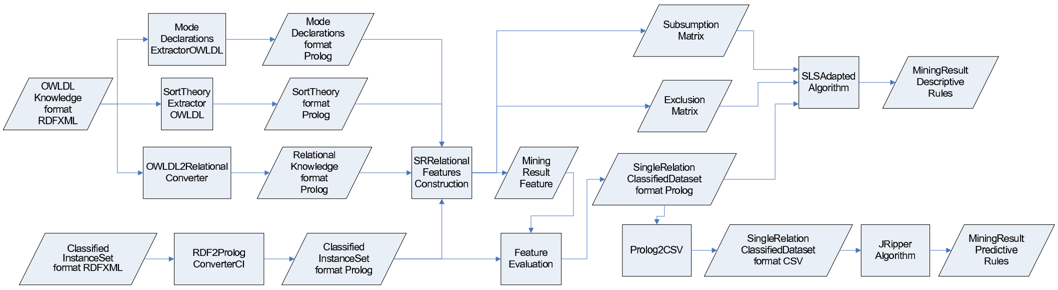
Mode Declarations ExtractorOWLDL
Mode Declarations format Prolog
Subsumption Matrix
OWLDL Knowledge format RDFXML
SortTheory Extractor OWLDL
SortTheory format Prolog
SLSAdapted Algorithm
MiningResult Descriptive Rules
Exclusion Matrix
OWLDL2Relational Converter
Relational Knowledge format Prolog
SRRelational Features Construction
Mining Result Feature
SingleRelation ClassifiedDataset format Prolog
Classified InstanceSet format RDFXML
RDF2Prolog ConverterCl
Classified InstanceSet format Prolog
Feature Evaluation
Prolog2CSV
SingleRelation ClassifiedDataset format CSV
JRipper Algorithm
MiningResult Predictive Rules

Figure 2. An automatically generated workflow for obtaining classification (predictive) rules and subgroup descriptions (descriptive rules) from annotations of CAD design drawings.

References

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (2003). *The description logic handbook, theory, implementation and applications.* Cambridge University Press.

Bernstein, A., & Deanzer, M. (2007). The next system: Towards true dynamic adaptions of semantic web service compositions (system description). *Proceedings of the 4th European Semantic Web Conference (ESWC'07).* Springer.

Blum, A., & Furst, M. (1997). Fast planning through planning graph analysis. *Artificial intelligence*, *90*, 281–300.

Brezany, P., Janciak, I., & Tjoa, A. M. (2007). Ontology-based construction of grid data mining workflows. *Data Mining with Ontologies: Implementations, Findings and Frameworks.* IGI Global.

Deelman, E., Blythe, J., Gill, Y., Kesselman, C., Koranda, S., Lazzarini, A., Mehta, G., Papa, M., & Vahi, K. (1990). Pegasus and the pulsar search: From metadata to execution on the grid. *Parallel Processing and Applied Mathematics.* Springer.

Dzeroski, S. (2007). Towards a general framework for data mining. *Knowledge Discovery in Inductive Databases - 5th International Workshop, KDID'06, Revised, Selected and Invited Papers* (pp. 259–300).

Hoffmann, J., & Nebel, B. (2001). Online convex programming and generalized infinitesimal gradient ascent. *Journal of Artificial Intelligence research*, *14*, 253–302.

Klusch, M., Gerber, A., & Schmidt, M. (2005). Semantic web service composition planning with owls-xplan. *1st Intl. AAAI Fall Symposium on Agents and the Semantic Web.*

Mannila, H. (1995). Aspects of data mining. *Procs of the MLnet Familiarization Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases.*

Morik, K., & Scholz, M. (2004). The miningmart approach to knowledge discovery in databases. *Proceedings of the International Conference on Machine Learning* (pp. 47–65). Springer.

Patel-Schneider, P., Hayes, P., & Horrocks, I. (2004). Owl web ontology language semantics and abstract syntax.

Smith, D., & Weld, D. (1999). Temporal planning with mutual exclusion reasoning. *Proceedings of the 1999 International Joint Conference on Artificial Intelligence (IJCAI-1999)* (pp. 326–333).

Suyama, A., Negishi, N., & Yamagchi, T. (1998). Composing inductive applications using ontologies for machine learning. *Proceedings of the First International Conference on Discovery Science* (pp. 429 – 431).

Trajkovski, I., Zelezny, F., Lavrac, N., & Tolar, J. (2008). Learning relational descriptions of differentially expressed gene groups. *IEEE Trans. Sys Man Cyb C*, *38(1)*, 16–25.

Žáková, M., & Železný, F. (2007). Exploiting term, predicate, and feature taxonomies in propositionalization and propositional rule learning. *ECML 2007: 18th European Conference on Machine Learning.*

Žaková, M., Železný, F., Garcia-Sedano, J. A., Massia-Tissot, C., Lavrač, N., Křemen, P., & Molina, J. (2006). Relational data mining applied to virtual engineering of product designs. *Procs of the 16th Int. Conference on Inductive Logic Programming* (pp. 439–453). Springer.

Wirth, R., Shearer, C., Grimmer, U., Reinartz, T. P., Schloesser, J., Breitner, C., Engels, R., & Lindner, G. (1997). Towards process-oriented tool support for knowledge discovery in databases. *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery* (pp. 243 – 253).

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Yang, Q., & Wu, X. (2006). 10 challenging problems in data mining research. *Intl. Jrnl. of Information Technology and Decision Making*, *5(4)*, 597–604.

Selecting Classifiers Using Meta-Learning with Sampling Landmarks and Data Characterization

Rui Leite RLEITE@LIAAD.UP.PT
Pavel Brazdil PBRAZDIL@LIAAD.UP.PT
LIAAD-INESC Porto L.A./Faculty of Economics,
University of Porto, Rua de Ceuta, 118-6,
4050-190 Porto, Portugal

Abstract

This paper concerns the problem of predicting the relative performance of classification algorithms. Previous approaches involved the concept of metalearning based on past experiments. The key idea was to learn the mapping between the characterization made on both the new dataset and the past ones and the actual performance observed. Several ways of characterizing the datasets were developed, some were data-based while others were algorithm-based (sampling landmarks). Although recent studies showed some advantages of the sampling landmark measures, data characterization can still be useful. A combination of both types of characterization is presented in this paper. We also present a method that iteratively selects the most appropriate measures for predicting the relative performance of classifiers. Experimental evaluation has shown that the method achieves better performance than previous approaches.

1. Introduction

The problem of predicting the relative performance of classification algorithms continues to be an issue of both theoretical and practical interest. There are many algorithms that can be used on any given problem. Although the user can make a direct comparison between the considered algorithms for any given problem using a cross-validation evaluation, it is desirable to avoid this, as the computational costs are significant.

The common approach of many methods is to store previous experimental results on different datasets. The datasets, including the one in question, are characterized using a set of measures. A (meta-)learning method is used to generate a prediction, for instance, in the form of a relative ordering of the algorithms.

Some methods rely on dataset characteristics such as statistical and information-theory measures (D. Michie, 1994; Brazdil et al., 2003). However, these measures need to be identified beforehand, which is a non-trivial task.

These difficulties have led some researchers to explore alternative ways to achieve the same goal. Some have proposed to use performance of simplified versions of the algorithms. These performance measures are sometimes referred to as *sampling landmarks* (Bensussan & Giraud-Carrier, 2000; Pfahringer et al., 2000).

Other researchers have proposed to use the algorithms performance on simplified versions of the data. These measures are sometimes referred to as *sampling landmarks* (Soares et al., 2001; Fürnkranz & Petrak, 2001).

The use of previous information concerning a series of sampling landmarks (i.e., learning curves) on a set of representative datasets is essential. Without this, sampling may lead to poor results (Perlich et al., 2003).

Our previous studies involving sampling landmarks have shown (Leite & Brazdil, 2005; Leite & Brazdil, 2007a; Leite & Brazdil, 2007b) that this approach performs usually better than *dataset characteristics* for predicting the relative performance of classifiers. The characterization involves conducting some experiments on a new dataset. The plan of these experiments is built up gradually, by taking into account the results of all previous experiments – both on other datasets and on the new dataset obtained so far.

We have observed that in some cases data characterization measures perform better. A question then arises as to whether it is possible to devise a method to predict the relative performance of classifiers using measures of both kinds. The aim of this paper is to answer this question. In the next sections we describe methods that use performance on samples (AMDS and ADMS_SetGen), data characteristics (MDC) and both types of measures (AMDS_DC).

2. Using Performance on Samples to Predict the Outcome of Learning

The aim is to decide which of two classification algorithms (*Ap* and *Aq*) is better on a new dataset *d*. The performance of the algorithms on a set of samples sheds some light on the final outcome of the learning curves. Intuitively this approach should work better if several samples of different sizes are used for each algorithm, providing more information about the shape of the two learning curves. Method AMDS (Leite & Brazdil, 2007a) uses this approach. It assumes the existence of available information about how both algorithms perform on different datasets $d_1 \cdots d_n$ for several samples with sizes $s_1 \cdots s_m$.

The attributes used for the prediction (on the best algorithm), also known as metaattributes, are estimates of the performance of the algorithms on random samples of specific sizes $s_1 \cdots s_m$. For instance metaattribute $Ap_{i,j}$ ($Aq_{i,j}$) is the estimate of performance of algorithm *Ap* (*Aq*) on a random sample of size s_j extracted from dataset d_i. Normally only some of these metaattributes will be used in the prediction. In other words, only a parts of each learning curve will be used. The user can either specify which metaattributes should be used, or else we can use an automatic mechanism for selecting them (in Section 4 we describe the method used to determine the appropriate metaattributes).

Assuming that method AMDS will use a fixed set of metaattributes S (determined by the user), the method AMDS encompasses the following steps:

1. Characterize the new dataset d (involves the estimation of all metaattributes in set S)
2. Compute the distances between d and all the other datasets $d_1 \cdots d_n$ using the description given by the metaattributes S. Identify the subset of k nearest datasets.
3. For each of the k nearest datasets identified (retrieved) in step 2, *adapt* each pair [1] the other withof learning curves to the new partial learning curves build for dataset *d*. Adaptation is done by rescaling each retrieved learning curve in order to minimize the square distance from this curve to the respective partial learning curve for dataset *d*.
4. For each pair of adapted curves decide which algorithm is better.
5. Identify the algorithm to use on the new dataset *d*, by considering the results on k neighbour datasets.

It is clear from the overview that AMDS is a k-NN classifier. It can be seen as a meta-level classifier, whose aim is to determine which base-level algorithm (*Ap* or *Aq*) is better. The distance function [2] is given by the following equation:

$$d_{AMDS}(d_i, d_j) = \underbrace{\sum_{s \in S_p} \mid Ap_{i,s} - Ap_{j,s} \mid}_{Ap} + \underbrace{\sum_{s \in S_q} \mid Aq_{i,s} - Aq_{j,s} \mid}_{Aq} \qquad (1)$$

S_p (S_q) contains the indices of meta-features used to characterize the new case (dataset) using the performance of algorithm *Ap* (*Aq*) on specific samples.

3. Using Data Characteristics to Predict the Outcome of Learning

This method (MDC) can be briefly described by:

1. Compute the characterization measures [3] for all datasets (including the new one).
2. Compute the distance between the new dataset and the stored ones.
3. Choose the k stored datasets (neighbours) that are "nearest" to the new dataset (according to the distance).
4. Use the algorithm that was most often the best one on the identified nearest datasets.

A Manhattan distance function is used to identify the nearest datasets. Each measure is first rescaled to fit the interval [0,1]. The distance function is:

$$d_{MDC}(d_i, d_j) = \sum_{t \in M} \frac{\mid M_t(i) - M_t(j) \mid}{max(M_t) - min(M_t)} \qquad (2)$$

[1] Each dataset has associated a pair of learning curves, one related with *Ap* and other with *Aq*.

[2] This is distance is usually called Manhattan distance.

[3] We have considered the data characteristic measures presented in Table 1. Method MDC could use other measures.

4. Selecting which Measures to Use (SetGen)

Method AMDS predicts the relative performance of classification algorithms using their performance on different samples. The sample sizes are passed to AMDS as an input (e.g. <s1,s2,s3> related to Ap and <s2,s5> related to Aq). It is important to find how many samples should be used and what their sizes should be. If we use the performance on more samples (or larger samples) it is reasonable to expect an improvement on the quality of the decisions made by AMDS (the learning curves shapes will be better defined). However the computational costs involved in computing the metaattributes will rise. If we use less samples (or smaller samples) both the quality of the decision and the computational costs will decrease.

Algorithm SetGen determines automatically the metaattributes. This is not dome using an ordinary feature selection method (e.g. forward selection) that only tries to improve the accuracy of the method. The method described here also deals with costs, that is time spent to compute each feature.

The samples are chosen taking into account the expected success rate improvement in the prediction concerning the algorithm to use (Ap or Aq). A new meta-feature can be included only if the success rate of AMDS is expected to improve by at least a fixed amount Δ. At the same time the system tries to identify those candidate solutions that increase the computational costs by the least amount. The desired sequence of samples is identified using a hill climbing approach. An overview of the method is presented in Figure 1. The best metafeature is the one with minimum estimated cost that will cause an expected improvement on the accuracy of AMDS by at least Δ.

Figure 2 shows the method SetGen in more detail. The *Simplify* instruction (step 20) removes from MF all metaattributes covered by the set of metaattributes constructed so far. If for instance $AtrSet$ contains metaattribute Ap_4 then all Ap_i for $i \leq 4$ will be removed from MF.

5. Using Performance on Samples and Data Characteristics to Predict the Outcome of Learning

The method AMDS_DC is an extension of the method AMDS. The difference relies on the distance function used. The distance function includes both estimates of the performance of the algorithms on samples and also data characteristic measures. The distance is defined

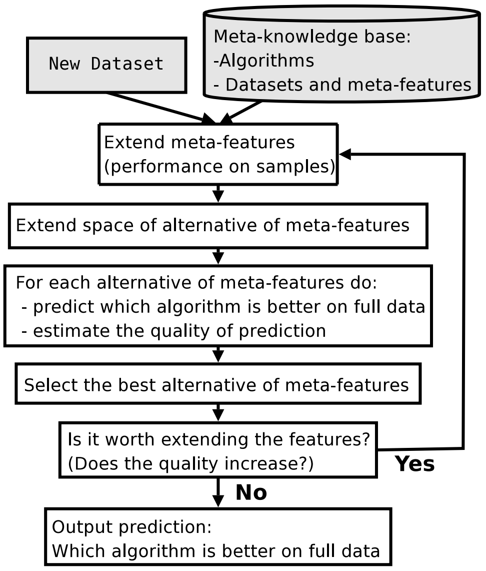
New Dataset
Meta-knowledge base:
-Algorithms
- Datasets and meta-features
Extend meta-features
(performance on samples)
Extend space of alternative of meta-features
For each alternative of meta-features do:
- predict which algorithm is better on full data
- estimate the quality of prediction
Select the best alternative of meta-features
Is it worth extending the features?
(Does the quality increase?)
Yes
No
Output prediction:
Which algorithm is better on full data

Figure 1. SetGen: An Iterative process of characterizing the new dataset and determining which algorithm is better

Function `SetGen(Δ)`

```
1  $MF \leftarrow$ Enumeration of all metaattributes
2  Estimate the computational cost of every meta–attribute
3  Sort $MF$ according to the computational costs
4  $Improved \leftarrow$ True
5  $AtrSet \leftarrow \{\}$
6  while $Improved$ do
7      $Improved \leftarrow$ False
8      $last\_acc \leftarrow$ EvalAMDS_DC($AtrSet$)
9      foreach $mfeat \in MF$ do
10         $last\_acc \leftarrow$ EvalAMDS_DC($AtrSet \cup \{mfeat\}$)
11         if $mfeat$ is a data characteristic then
12             $\delta \leftarrow 0.001$
13         else
14             $\delta \leftarrow \Delta$
15         end
16         if $acc > last\_acc + \delta$ then
17             $AtrSet \leftarrow AtrSet \cup \{mfeat\}$
18             $acc \leftarrow last\_acc$
19             $Improved \leftarrow$ True
20             Simplify($MF$, $AtrSet$)
21         end
22     end
23 end
24 return $AtrSet$
```

Figure 2. SetGen: Method that Select the Measures

by the following equation:

$$d(d_i, d_j) = d_{AMDS}(d_i, d_j) + d_{MDC}(d_i, d_j) \quad (3)$$

Since the data characteristic measures were previously shown to be less predictive in comparison with performance on samples metaattributes we have modified the criteria used to identify the best metaattribute. This type of metaattribute is introduced, if the accuracy increase is at least Δ_{DC}, which is a much smaller value than the value of Δ.

6. Empirical Evaluation

In our previous work (Leite & Brazdil, 2007b) we have shown that AMDS is better than MDC for predicting the best algorithm for a given dataset. The method AMDS has lower costs when compared with the decision using cross–validation, but larger costs when compared with MDC.

In this section we present the results concerning the method AMDS_DC. We are interested to know whether the data characteristics (using AMDS_DC) can improve the performance of our previous method AMDS. The question we address here is: Will SetGen select data characteristic measures in conjunction with the measures of performance of the algorithms on samples (sampling landmarks)?

Here we describe the experiments that we have conducted to evaluate the method AMDS_DC. Comparisons are made with MDC and AMDS (version that includes SetGen).

The decision problem concerning whether algorithm A_p is better than A_q can be seen as a classification problem which can have three different outcomes: 1 (or -1) if algorithm Ap gives significantly better (or worse) results than A_q, or 0 if they are not significantly different. The aim of this experimental study is to determine the accuracy and the computational cost (calculated at run-time) of the three methods considered.

To compute the accuracies we need to compare, for each case (dataset), the predicted class with the true classification determined by a usual cross–validation evaluation procedure on each dataset for the two given algorithms. A statistical test (t–test) was used to compute the statistical significance. Instead of using the usual accuracy measure, we have used a different measure that is more suited for our classification task with 3 possible outcomes. The errors are called *penalties* and are calculated as follows. If a particular method (e.g. AMDS) classifies some case as +1 (or -1), while the true class is 0 (the given algorithms are not significantly different) then, from a practical point of view the method did not fail, because any decision is a good one. Therefore the penalty is 0. However if the method classifies the dataset as 0, while the true class is +1 (or -1) then we consider that the method partially failed. The penalty is 0.5. If the classification is +1 (or -1), while the true class is -1 (or +1), this counts as a complete failure, and the penalty is 1. The corresponding accuracy, referred to as *meta–accuracy*, is computed using this formula $1 - \sum_{d \in D} \frac{penalty(d)}{|D|}$, where D is the collection of datasets.

In this empirical study we have used the following 6 base-level classification algorithms, all implemented within Weka (Witten et al., 1999) machine learning tools: J48 (C4.5 implemented in Weka), JRip - rule set learner (RIPPER (Cohen, 1995)), logistic discriminant (le Cessie & van Houwelingen, 1992), MLP - multi-layer perceptron, IB1 - instance-based learner, and Naive Bayes. Using this setting we get 15 classification problems, one for each pair of algorithms.

We have used 40 datasets in the evaluation. Some come from UCI (Asuncion & Newman, 2007), others from the project METAL (MetaL, 1999). In our experiments described here we have considered the data characteristic measures presented in Table 1.

Table 1. Data Characteristics Measures

	Id	Measure
1	M1	n.examples
2	M2	prop.symbolic.attrs
3	M3	prop.missing.values
4	M4	prop.h.outlier
5	M5	class.entropy
6	M6	avg.mutual.information
7	M7	canonical.correlation.best.linear.combination

For each decision problem (there are 15 such problems [4]) the three methods (AMDS_DC, AMDS, MDC) are evaluated using a leave-one-out methodology. In each case we measure meta-accuracy and computational cost. Computational costs are expressed as a ratio of times. The time required by the particular method (e.g. AMDS_DC) is normalized, by dividing it by the time required to obtain the decision by cross-validation. Cross-validation represents a slower, but, in principle, also a more reliable method.

In the experiments the following setting were used $\Delta = 0.07$, $\Delta_{DC} = 0.001$ and $k = 23$ for AMDS, AMDS_DC, and $k = 7$ for MDC.

Table 2 shows the results concerning the meta–

[3] For 6 items, we can define 15 different pairs.

accuracies. The decision problem in question is presented in column 1. In most cases both AMDS and AMDS_DC are better than MDC. There are some cases where MDC is worse than the default accuracy. In most cases AMDS_DC reaches the same performance as AMDS. This is due to the fact that usually the performance on samples is more predictive than data characteristic measures. Most of the times the measures based on characteristics are not selected by SetGen and when they are selected the decision concerning the best algorithm remains the same. However in problem IB1–LOG AMDS_DC is better than AMDS.

Although the results did not improve significantly the performance of the previous method, AMDS_DC is never worse results than AMDS. This indicates that we can use AMDS_DC.

Table 2. Meta–accuracy for each method (average)

	Default Accuracy	MDC	AMDS	AMDS_DC
IB1-J48	80.00	**95.00**	92.50	92.50
IB1-JRip	72.50	87.50	**92.50**	**92.50**
IB1-LOG	66.67	80.56	94.44	**97.22**
IB1-MLP	84.21	85.53	**92.11**	**92.11**
IB1-NB	67.50	70.00	**92.50**	**92.50**
J48-JRip	75.00	76.25	**90.00**	**90.00**
J48-LOG	77.78	70.83	**97.22**	**97.22**
J48-MLP	63.16	75.00	**86.84**	**86.84**
J48-NB	85.00	76.25	**90.00**	**90.00**
JRip-LOG	75.00	72.22	**86.11**	**86.11**
JRip-MLP	68.42	81.58	**89.47**	**89.47**
JRip-NB	82.50	78.75	**92.50**	**92.50**
LOG-MLP	80.00	78.57	**97.14**	**97.14**
LOG-NB	**94.44**	**94.44**	**94.44**	**94.44**
MLP-NB	**94.74**	**94.74**	**94.74**	**94.74**
Mean	77.79	81.15	92.17	**92.35**

Regarding the costs of the methods, that is measured by the time spent on computing required metafeatures, the results are summarized in Table 3. The results are presented in the form of ratios. The time spent by each method is related to the time taken by cross-validation. The values presented here are average values (geometric mean). MDC is the fastest method (however its performance is worse that AMDS and AMDS_DC). It is 143 times faster that cross-validation. AMDS_DC and AMDS are 7 times faster than cross validation.

Table 3. Computational costs in comparison with cross-validation (ratios)

	MDC	AMDS	AMDS_DC
Geometric mean	0.007	0.137	0.139

A question arises regarding which data characteristic measures were selected by SetGen when running AMDS_DC. Table 4 presents the results. The numbers shown indicate in how many datasets each particular measure was introduced. The numbers are relative. They show the proportion of datasets in which a particular measure was used. Number 2.5, for instance, means that the feature was introduced in 1/40 dataset (the total number of datasets was 40). For some decision problems SetGen did not select any measure.

Table 4. Percentage of use of each measure

	M1	M2	M3	M4	M5	M6	M7
IB1-J48							12.5
IB1-JRip	7.5						
IB1-LOG	**2.8**		**5.6**		**2.8**		
IB1-MLP							
IB1-NB				2.5	2.5	2.5	
J48-JRip							
J48-LOG		2.8				2.8	
J48-MLP			2.6	2.6		18.4	5.3
J48-NB							
JRip-LOG		2.8		2.8		2.8	
JRip-MLP		2.6		13.2			
JRip-NB							
LOG-MLP							
LOG-NB							
MLP-NB							

7. Discussion

How is the method described related to planning?

In this section we will discuss the issue of how the method presented in this article relates to planning. We start by reviewing the area of planning and execution control.

Planning has been an active area of research already in the early days of AI. The topic of planning is discussed in just any textbook on AI (see e.g. (Russell & Norvig, 2003)). The early approaches (e.g. in the 70's) exploited usually a symbolic representation of states, goals and the operators. Plans were represented usually as sequences of operations, or by a partially ordered set of operations (referred to usually as non-linear plans). Later approaches exploited hierarchical decomposition, conditional planning and integration of planning and execution (Russell & Norvig, 2003).

Approaches to planning can be divided into two groups depending on whether on what the objective is. The first group include planners that generate a full plan which is then followed step by step in the execution phase. If some problem is detected during execution, the system then corrects the plan (preferably by carrying out a small modification, but not by complete re-planning).

The second approach just determines the first action (or a few actions) to execute. The rest of the plan is not fully determined. The system then executes the actions and continues in this mode until the final goal has been attained. The latter approach is useful in situations where there is incomplete knowledge of the environment and when the execution of some actions brings in more information. Consider, for instance, that the robot has the task of retrieving an object from a dark room. The robot should start by entering the room, localizing a switch and turning the light on. It makes no sense to do any other planning before it can see the layout of the room. After this, when more information is available, the robot can plan how to get to the required object.

The method presented here follows the second approach referred to above. It tries to establish the next action to execute and executes this action. It continues like this until the stopping condition is satisfied. In our case the aim of the action it to gather more information about one of the two algorithms in question. This involves training each algorithm on a sample of a particular size and getting the estimates of performance on a test sample. This process terminates when enough information is available and a decision can be made as to which of two algorithms is better.

What is special about our planning problem?

The problem tackled here is a somewhat different from typical problems handled by many planning systems. Let us consider the nature of the goal in learning a classifier. Note that the problem does not involve an issue of how to solve a particular task, but rather a set of tasks. This is because the dataset in question represents, in effect, a set of classification problems. This is because, in attribute-based representation, each line represents one classification problem. Each solution is characterized by a particular performance measure (e.g. accuracy).

This domain has another somewhat special characteristic. Solutions are characterized by training and test times, or in general by costs. So normally, considering that many algorithms exist, these will have different benefits (e.g. accuracies) and different costs. So, on one hand we can identify a low cost solutions (fast algorithms) which normally do not perform very well. On the other hand we may have solutions with achieve higher performance, but incur also higher costs. In the work presented here we were not interested in the extremes, but rather a good compromise between the two. Our aim is to obtain a solution that approaches the performance of the best possible solution (selection by cross-validation), but incurs much lower costs (i.e., is N times faster).

Is it better to recover complete plans or recover information and re-plan?

In some earlier work on planning DM operations, there exists a notion of a plan associated with a given task, which normally consists of a partially ordered set of operations. If these are stored, they can be re-used in new setting. It is just necessary to identify the task that is similar to the current task, retrieve the plan and adapt it to the new circumstances. This is the approach followed in MiningMart (Kietz et al., 2000), for instance. A question arises about how the approach described in this article relates to this.

In this article we have described a method that suggests a sequence of experiments on samples of certain size. Training a particular classification algorithm can indeed be seen as an operation. Elaborating a suitable sequence of such operations can thus be regarded as planning.

After the planning process has been completed, the complete plan is available for inspection or for future use and/or adaptation. Note that we do not just retrieve a plan and re-use it, but rather exploit previous information to generate a plan that is well adapted to the new circumstances.

Retrieving and existing plan and executing it (i.e without adaptation), is in principle, possible, but we have not done that, as it would most likely lead to substandard results. We plan to verify this experimentally in future.

8. Conclusions

In this paper we have described method AMDS_DC that exploits data characterization and sampling to determine which of two given classification algorithms is better on a new dataset.

Previous results have shown that method AMDS that exploits sampling landmarks, that is, metafeatures representing estimates of performance of given classification algorithms on samples, achieves more accurate decisions when compared with the method MDC that exploits data characteristics.

Here we have shown how we can extend our previous method, by including the measures of both kinds. The metafeatures (metaattributes) used by the method are either data characteristics of the dataset in question, or performance estimates of the given classification algorithms on samples of specified sizes.

The method automatically establishes how many samples are needed and their sizes. Besides the method also determines which data characteristic measures are useful for the decision concerning which is the best algorithm.

Experimental evaluation has shown that the new method (AMDS_DC) achieves a comparable performance to the previous method (AMDS). The average metaaccuracy was 92.35%. In one specific problem (IB1 *versus* LOG) AMDS_DC achieved a better result when compared with AMDS.

The method determines which metaattributes to use in order to reach accurate decisions while at the same time tries to save computational time. This represents a significant improvement over previous methods in dealing with the problem of predicting the relative performance of learning algorithms in a systematic manner.

References

Asuncion, A., & Newman, D. (2007). UCI machine learning repository.

Bensussan, H., & Giraud-Carrier, C. (2000). Discovering task neighbourhoods through landmark learning performances. *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD2000)* (pp. 325–330). Springer.

Brazdil, P., Soares, C., & Costa, J. (2003). Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning, 50*, 251–277.

Cohen, W. W. (1995). Fast effective rule induction. *Proc. of the 12th International Conference on Machine Learning* (pp. 115–123). Tahoe City, CA: Morgan Kaufmann.

D. Michie, D. e. C. (1994). *Machine learning, neural and statistical classification.* Ellis Horwood.

Fürnkranz, J., & Petrak, J. (2001). An evaluation of landmarking variants. *Proceedings of the ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-2001)* (pp. 57–68). Springer.

Kietz, J.-U., Vaduva, A., & Zücker, R. (2000). Mining-mart: Combining case-based-reasoning and multi-strategy learning into a framework to reuse kdd-application. *In: R.S. Michalki and P. Brazdil (editors), Fifth International Workshop on Multistrategy Learning (MSL2000), Guimares, Portugal, 2000.*

le Cessie, S., & van Houwelingen, J. (1992). Ridge estimators in logistic regression. *Applied Statistics, 41*, 191–201.

Leite, R., & Brazdil, P. (2005). Predicting relative performance of classifiers from samples. *ICML '05: Proceedings of the 22nd international conference on Machine learning* (pp. 497–503). New York, NY, USA: ACM Press.

Leite, R., & Brazdil, P. (2007a). An iterative process for building learning curves and predicting relative performance of classifiers. *Proceedings of the 2nd General Artificial Intelligence Workshop (GAIW 2007), in the 13th Portuguese Conference on Artificial Intelligence (EPIA 2007)* (pp. 87–98). Guimarães, Portugal: Springer.

Leite, R., & Brazdil, P. (2007b). An iterative process of building learning curves and predicting relative performance of classifiers. *Proceedings of the Planning to Learn Workshop (PlanLearn 2007), held in ECML/ PKDD 2007* (pp. 31–40). Warsaw, Poland.

MetaL (1999). Metal project site. http://www.metal-kdd.org/.

Perlich, C., Provost, F., & Simonoff, J. S. (2003). Tree induction vs. logistic regression: a learning-curve analysis. *J. Mach. Learn. Res., 4*, 211–255.

Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. (2000). Meta-learning by landmarking various learning algorithms. *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)* (pp. 743–750). Stanford, CA.

Russell, S. J., & Norvig, P. (2003). *Artificial intelligence: A modern approach.* Pearson Education.

Soares, C., Petrak, J., & Brazdil, P. (2001). Sampling-based relative landmarks: Systematically test-driving algorithms before choosing. *Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA 2001)* (pp. 88–94). Springer.

Witten, I., Frank, E., Trigg, L., Hall, M., Holmes, G., & Cunningham, S. (1999). Weka: Practical machine learning tools and techniques with java implementations.

Learning to Design Complex Systems Using Frequent Graph Patterns

José Ignacio Estévez IESTEVEZ@ULL.ES
Pedro Toledo PEDRO@ISAATC.ULL.ES
José Sigut SIGUT@ISAATC.ULL.ES
Silvia Alayón SILVIA@ISAATC.ULL.ES
Department of Systems Engineering and Control, University of La Laguna, 38200, S/C de Tenerife, Spain

Abstract

The problem of learning to design complex systems building networks of simpler components is tackled. A framework for the objective comparison of different learning strategies is proposed. The Workflow Schema is the data structure proposed for the problem formalization. Moreover, the use of graph mining results to define the learning process is analyzed and the advantages and limitations emphasized. Finally, an empirical analysis of the strategies and heuristics proposed in the defined benchmark framework is made, and the results shown.

1. Introduction

The problem of learning to design complex systems have been largely studied and it is still a challenging problem. The design of a complex system is usually understood as the construction of a network of simpler components, being the structure of the network itself the solution to the problem. We have focused in the frequent scenario in which the learning system has access to previous solutions. The information available might be used assuming the hypothesis that the frequent patterns in previous solutions should be suitable to be applied in new solutions to similar problems. This paper proposes a benchmark problem to compare objectively different approaches to the learning task. A strategy to apply graph mining results in learning strategies is then discussed. The simpler approach of using the frequency of appearance of isolated elements of the network in previous solutions is presented as well. The results achieved applying these strategies are compared, pointing up the difficulties to scale up the graph pattern approach to large sparse search spaces.

The described approach is not an attempt to propose a model to deal with the design learning problem in real scenarios, but a context in which it can be measured the benefits and limitations of different learning strategies proposals.

2. Workflow Schema Representation

Multiple formalisms and languages have been proposed in the literature for knowledge representation in this context. There is active research in this way, from fields closely related to the topic, being the Induction of Process Models from log data one of the most outstanding examples (Rozinat et al., 2007). Event-Driven Process Chains (EPC) are a widely used technique for modelling business processes. Petri Nets (Gyapay & Pataricza, 2003) and P-Graphs have been used in the context of Process Network Synthesis to give support to the optimization process (Friedler et al.,). The Business Process Execution Language (BPEL) standardized by OASIS, and YAWL are languages successfully implemented in BPM software solutions. Process Algebra and Finite State Machines are also valid formalisms used in this context.

Nevertheless, the model adopted in this paper is Workflow Schema (Greco et al., 2005), since it have been demonstrated to be expressive enough to represent structured solutions to design problems.

The structure of this model is a Directed Acyclic Graph (DAG), with constraints. The structure is the representation of the knowledge of the problem and fixes all the possible solutions that can be constructed to solve it. On this graph the nodes represent ac-

Appearing in *Planning to Learn Workshop, International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

tions that transform the input materials into the output products. The edges represent the possibility of bringing the output product of one action to another action that would take it as input material. Depending on the context of the problem, the materials and products may be physical objects, if we are trying to optimize the production cycle of an industry, or any other concept as a data structure, if the problem is related with the optimization of a Business Management Process. The WS can be considered as a metagraph, being those subgraphs that fulfill all the constraints possible solutions to the represented problem. The constraints exist since to realize the actions some minimum input material might be necessary, the amount of output product is limited, and the amount of product that can be brought to other activity is fixed. Furthermore, the process is not considered optimized if there is some product available that can be resused but it is not. This paper assume the context in which such a WS is available for the problem, or can be induced using background knowledge and rules (Rozinat et al., 2007). Each possible solution according to this structure is a subgraph called Instance. The previous knowledge about previous solution to a problem is called Valuated Instance-Set. The following definitions formalize these concepts.

A **Workflow Schema** W is a tuple $(V, E, V_o, V_f, n, p, c)$, where V is an ordered set of nodes, E is a set of edges defined as $E \subseteq \{(v_i, v_j) \mid i < j \wedge v_i \in (V \setminus V_f) \wedge v_j(V \setminus V_o)\}$, V_o is the set of start nodes, V_f is the set of end nodes, $n : (V \setminus V_o) \to \mathbb{Z}^+$ is the Node necessity function, $p : (V \setminus V_f) \to \mathbb{Z}^+$ is the Node production function, and $c : E \to \mathbb{Z}^+$ is the Edge capacity function.

Given a node $v \in V$ the set of input edges is denoted as $E^i(v) = \{(v', v) \mid (v', v) \in E\}$, and the set of output edges as $E^o(v) = \{(v, v') \mid (v, v') \in E\}$.

Given a $WS = (V, E, V_o, V_f, n, p, c)$, a graph $S = (V_S, E_S)$ is said to be a subgraph of WS iff (i) $V_S \subseteq V$ (ii) $E_S \subseteq E$ and (iii) $(\forall\ v_i \in V_S, \forall\ v_j \in V_S, i < j \to \exists\ path(v_i, v_j))$.

Given a WS, the subgraph $S_I = (V_I, E_I)$ is an instance of W, iff the following constraints are fulfilled:

(i) At least one of the initial nodes are included in the instance: $V_{Io} = V_I \cap V_o \neq \emptyset$, (ii), only the initial nodes have not ingoing edges: $\forall v_i \in V_I \setminus V_{Io}, \exists (v_j, v_i) \in E_I$ (iii) the sum of input edge capacities is at least the necessity of each node: $\forall v \in V_I \setminus V_{Io}, \sum\limits_{E_I^i(v)} c(e) \geq n(v)$, and (iv) given a node with at least one outgoing edge, its output edges are included in the instance with the goal of approximating as much as possible the node production to the sum of edge capacities, under the constraint that this sum of capacities can not be greater than the production:

$$\forall v, (v, v') \in E_I \to min_{\overline{E_I^o}(v)}\{c(e)\} > p(v) - \sum_{E_I^o(v)} c(e) \geq 0$$

where $\overline{E_I^o}(v) = E^o(v) \setminus E_I^o(v)$.

An instance I is complete when $\nexists v \in V_I \setminus V_f$ with $E_I^o(v) = \emptyset$.

A complete instance I is said to be **successfull**, SC-Instance, if it contains at least one final node, $V_{If} = V_I \cap V_f \neq \emptyset$,

An instance I defined respect to W, may be also defined as the join of a set V_{Io} of initial nodes of W, and a set of decisions D. A **decision** $D_I(v)$ of a node v is defined as the part of an instance constituted by the set of all the outgoing edges $E^o(v)$ included in the instance, and the set of the end nodes of these outgoing edges.

An instance I' represented by the subgraph S'_I is an extension of another instance I represented by the subgraph S_I, $I \prec I'$, iff I' and I are defined respect to the same WS and S_I is a subgraph of S'_I.

Given a non-complete Instance I, an extension I' is said to be a 1-step extension iff I' can be obtained joining I with at least one decision of I'. This is $I \underset{1}{\prec} I'$ `iff` $(I \neq I') \wedge (\exists v \in V_I \mid I \cup D_{I'}(v) = I')$. By analogy, a extension I^n of an instance I is said to be a n-step extension iff I^n is obtained as the union of I with at least n decisions of I^n

Given a WS, and a SC-Instance $\mathbf{I} = (V_I, E_I)$ defined respect to it, the cost function associated to that instance can be generally defined as:
$Cv : InstancesSet \to \mathbb{R}$. Therefore, the cost function assigns a real value to each combination of nodes and edges that fulfills SC-Instance constraints, without any additional information or assumption.

The set of all possible different SC-Instances defined respect to a given WS is called $2^{I_{WS}}$. The cardinality of this set, $\mid 2^{I_{WS}} \mid$, is the size of the solution space modelled by the WS.

Given a WS, a Valuated Instance-Set $\mathbf{F}$ is defined as a set of pairs:
$F \subseteq \left\{(I_i, C_i) \mid I_i \in 2^{I_W S} \wedge C_i = Cv\left(V_{I_i}, E_{I_i}\right)\right\}$.

3. Benchmark Problem

The former definitions give the context in which the Benchmark Problem can be defined. The purpose is not to propose a schema to be immediately applied in real world situations, but a framework that allows to compare objectively different learning strategies.

The definition of this framework assumes given: a Workflow Schema **WS**, a Valuated Instance-Set **F** and a non-complete Instance I. The known information from the Cost function **Cv**, are the sampled values for the instances of the set F. Nevertheless, this function Cv can be used to analyze the behavior of the compared learning strategies.

With this information the objective is to achieve one of the following goals: (i) Design the system able to learn from the Workflow Schema and the Valuated Instance-Set to predict the optimum Instance I', extension of a given I for each possible non complete instance defined respect to the WS. This is, the learner might be a function $PL\left(WS,F\right) = IP$, where PL is the Predictor-Learning strategy function and the IP the Instance-Predictor induced function. The IP function is applied $IP(I) = I'$, being I a non-complete Instance and I' the optimum predicted complete Instance extension to I. (ii) Design a system able to learn a function that given the non-complete instance I predicts the optimum 1-step extension instance I'. In this case, the learner is a function $EL\left(WS,F\right) = IE$, where EL is the Extender-Learning strategy function and the IE the Instance Extender induced function. The function IE is applied $IE(I) = I^{+}$, being I^{+} the 1-step extension of I.

3.1. Learner Brief Analysis

The defined framework allows different approximations of learning systems to be applied to solve the problem. A couple of examples are outlined below.

Naively, the proposed problem can be seen as a combinatorial problem (Rosen et al., 2000), having one decision to take between a finite number of options for each node of the WS. Nevertheless, in the problem appear constraints between the values for these decisions since not all possible combinations lead to valid solutions. This approach might also define the way of using efficiently the knowledge from previous solutions.

Since the set of nodes of a WS is ordered, the process of building a complete instance can follow it. At each step the next node of the non-complete instance is taken and one of the possible decisions is added to it. This system can not be considered a Markov Process in which the state is represented by the selected node at each moment, since the decisions selected from previous nodes changes the resulting Instance completely, and so, the decision criteria to select for the actual node. If the state of the Markov process is defined as a function of the nodes of the actual non-complete Instance, then this consideration can be assumed. Nevertheless, the number of different states needed to represent all the situations grows exponentially with the number of nodes of the non-complete Instance.

4. Experiments and Results

Within the framework described by the proposed Benchmark Problem, some experiments have been designed to show how can be measured the differences between different learning strategies.

With respect to the definition of the framework, in order to make the experiments, the cost function Cv, that would remain unknown for the learner, might be specified. In our case we have defined a weight function for each element of the WS: $w : V \cup E \rightarrow \mathbb{Z}^{+}$ and it is called the local cost function. This function simply assigns a cost value to each element of the structure of the WS. Finally, the cost of a SC-Instance I on a WS is calculated as: $Cv(I) = \sum\limits_{v \in V_I} w(v) + \sum\limits_{e \in E_I} w(e)$

4.1. Heuristics for the EL function

Our main objective in these experiments is to study how to exploit information from the set of complete instances. The defined EL-functions are defined stochastically. All the decisions of each node of the WS are evaluated. The probability that the function EL returns one particular decision for a particular node is proportional to the valuation of the decision.

Random choice With the random choice heuristic, the EL function selects each decision randomly using a uniform probability distribution.

Isolated element mining based choice In this heuristic, the EL function is updated using information about the appearance of isolated elements (nodes and edges) in previous instances. Note, that this heuristic is not using information about the workflow structure where the element is included.

A node $v_i \in R_v$ of the workflow is valuated in the context of the set of alternative decisions R_v that are present when an instance has to be extended from the particular node v. Two aspects are considered. Firstly, each element is valuated obtaining the mean cost of the instances where it is included. If v_i is the node, the value q' assigned to it, in this first step is given by:

$$q_i' = \frac{1}{|S_I^{v_i}|}\sum_{S_I^{v_i}} \frac{(c_{max} - C(I))}{c_{max} - c_{min}}$$

where $S_I^{v_i}$ is the set of previous instances where element v_i is present and c_{min}, c_{max} are the best and worst costs of the defined instances.

In the second step, the frequency of the elements in the instances is taken into consideration. Frequency is used to non-linearly increase the value assigned to a node if it has a good value, or to non-linearly decrease the value assigned to a node if it has a poor value. This is carried out using the A-law non linear formula:

$$Q(q;A) = \begin{cases} \frac{Aq}{(1+ln(A))} & -\frac{1}{A} < q < \frac{1}{A} \\ sign(q)\frac{1+ln(A|q|)}{1+ln(A)} & |q| \geq \frac{1}{A} \end{cases}$$

Then, the EL performs a random experiment where the outcome are elements of R_v. The experiments are repeated until a complete decision is obtained.

Frequent pattern mining based choice The objective now is to introduce in the EL function preferences for particular structures in the WS.

The levelwise search algorithm (Agrawal & Srikant, 1995; Greco et al., 2005) is used to obtain frequent patterns from the previous instances. Once these patterns have been obtained, they are evaluated according to the same procedure as in the previous heuristic for isolated elements.

For each decision parameter v for the EL function, each single element in R_v is valuated with the information of the patterns in which it appears. The set of patterns considered in the valuation of an element are selected from the set of frequent patterns with two important constraints: they must be compatible with the instance that is being extended when the EL function is applied and they have to include the element that is being evaluated. Furthermore, only representative patterns are be selected. To achieve this, a subset of the previous complete instances is extracted containing the N_b best valuated instances and the N_w worst valuated instances. Frequent patterns obtained from this subset are the ones used for element valuation.

4.2. Learning Framework Strategies

To compare the different EL heuristics mentioned, a framework might be defined. In this case it is inspired in an `Active Learning Context`. In this context the learner is given a WS, and an empty set F. A fixed number N of opportunities to sample the function C_v are established. After this N values, the learner might have built the elements of the set F for N different Instances.

The learners can be finally compared by the values and evolution of these N elements of the F set. The best strategies to build the learner might also lead to a quick evolution of the instances in F to low-costs ones, in the sense that it means that it is exploring the near-optimal areas of the solution space.

The different learning frameworks implemented for the experiments are described below. All the strategies need a first set of K complete instances to obtain knowledge from the WS. This set can be obtained using the random choice EL heuristic. After those K instances, the learning strategy might be defined. The strategies proposed in this paper are defined specifying two aspects: Each $N - K$ of the instances to be created during this stage of the learning process can be achieved by the process: A non-complete Instance is proposed and then extended to achieve a SC-Instance. (i) The learning strategy might decide how to select the non-complete Instance on each iteration. (ii) To obtain the SC-Instance from the non-complete one, the EL function heuristic might be selected.

In relation to the first aspect, we have researched two types of randomized search strategies: a global search with reinforcement learning and a local search with a constructive stage.

Global search with reinforcement learning This is the most direct way of using the set of valuated complete instances and the heuristics described above. After obtaining a first set of complete instances, the reinforcement learning strategy tries to construct new instances using the EL function guided by the element valuations. Once the new instance is finished its cost is obtained sampling the function Cv and the valuations of the elements in each R_v for $v \in V$ are changed using one of the heuristics to consider the new information.

Local search with a constructive stage. This strategy can be seen inside a mixed search procedure including cycles of global search and others of local search (Resende & Ribeiro, 2005). The global search can be implemented using the random choice heuristic or the previous strategy. The local search cycle follows these steps:

It starts selecting one instance from F. For this selection the probability of each instance is proportional to its cost.

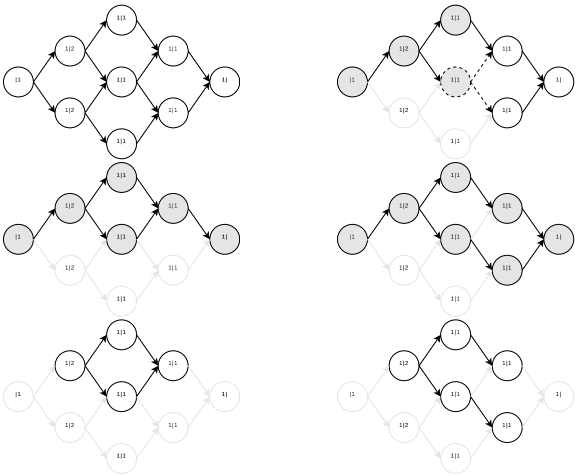

Figure 1. Workflow Schema, Instances and Patterns.

A frequent pattern is selected randomly. The probability to be chosen is also proportional to the pattern valuation.

In the constructive stage, a new instance is obtained from the selected instance and pattern. Firstly, the pre-pattern is obtained: it is a valid instance composed of the minimum subgraph of the instance that contains the selected pattern and the start node. Then the pre-pattern is extended using the EL function until a complete instance is obtained.

After the constructive stage, the new instance is valuated and included in the set of complete instances. A new iteration of the local cycle starts.

4.3. Workflow Example

Graph represented in figure 1 could be part of a workflow schema. Necessity and production values are written inside each node. All the node costs are equal. All the edges capacities are equal to one.

The second graph represented in figure 1 shows one instance on this WS. The nodes that belongs to the instance have been gray colored. Note that this instance can be extended making a decision on the elements drawn using dashed lines. Lower part of the figure shows the possible instances that result from extension. It is clear, that the first one has a lower associated cost than the second instance.

Observe that information needed to decide in a problem like this can not be obtained from isolated elements valuations.

The patterns shown in the bottom of figure 1 can be obtained from the proposed initial instance.

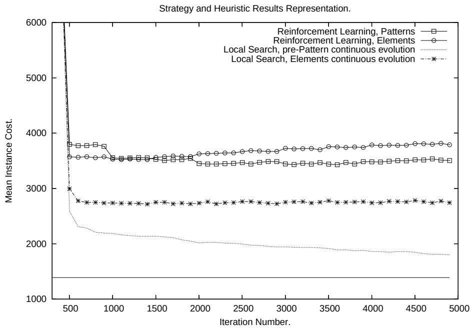

Figure 2. Randomized search heuristics results.

4.4. Implementation and results

For the implementation of the experiments, some assumptions have been made, as explained below. The experiments described in this paper are restricted to WS's with only one start node and only one end node ($|V_o| = |V_f| = 1$). The other important constraint is that every instance I of our WS can be extended to a complete instance.

The generated WS has 9 nodes, 45 edges and the number of different complete instances defined on it is about 40000. The cost of the optimal instance is 1389.

Figure 2 represents the results of the experiments making use of different search strategies and heuristics. Each strategy performs a number of identical experiments. Each experiment generates 5000 instances following the strategy rules. The experiments are repeated 100 times and the cost of each instance on each experiment stored. In order to know the general behavior of the strategy the mean over this 100 experiments is calculated. So that, a sequence of 5000 values is obtained and each of them represents the mean cost of the generated instances at the iteration number n. Then, a new sequence is calculated with each value representing the mean of 100 consecutive values in the sequence.

Each strategy begins constructing 500 random instances. After these iterations, the heuristics are applied.

The reinforcement learning with pattern information strategy is able to use more information from the previous instances than the strategy based on elements information. This result is clearly represented in figure 2. After some iterations in which there are not important differences, the strategy using frequent pat-

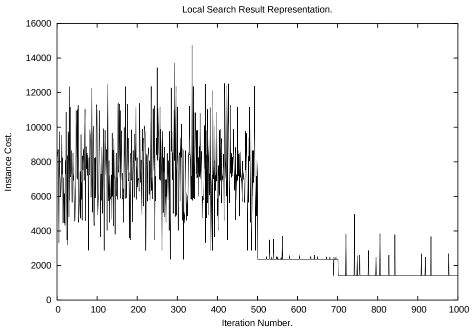

Figure 3. Local Search Experiment Detail.

tern mining begins to generate instances with lower costs than the other one. As it is shown, these two strategies have a problem in common: they are not able to improve the mean cost of the iterations they generate after the random period. The reason is that in the first period is useful to get information not only about instances well valuated but also about the worst instances. This strategy performs a global search in a search space very large and complex. The search process is not able to cover relevant parts of the workflow and relevant patterns are poorly or not represented at all in the set of frequent patterns.

Figure 2 also represents the results of the local search strategy, using element mining heuristic and pattern mining heuristics. It can be observed that the former results are much worse than latter ones. When using element mining heuristics, after some iterations the new instances generated are not better than the previous ones. On the other hand, the pattern mining heuristic results not only are better, but also they progress on each new instance generated.

The local search is represented in detail in figure 3. In this figure is shown the cost of a sequence of instances just after the random stage. In this case, the cost reduction process take place from the first iterations. After the random stage the best instances are used in the local search. When this search finds a better instance then this instance is used in the local search and the medium cost of the new instances constructed is reduced.

5. Conclusions

This paper has described a framework to compare objectively different learning approaches to the problem of the design of complex systems. The first contribution is the selection of the data representation structure between the different approximations proposed in the literature. This structure is a Workflow Schema based on a DAG with constraints. With this definition, the benchmark problem was defined, assuming that background knowledge from the application problem is available to build the structure of all problem with all possible solutions in form of a network structure. Some solutions to that problems might also be available and evaluated. Then, experiments are proposed. In these experiments some learning strategies are compared. It has been shown the utility of mining frequent patterns to train the learner, but also the limitations of this heuristics while searching a large sparse solution space. The exploration-exploitation trade off situating the experiment on a active learning like environment have been also faced. The results show that a local search around previously known close to optimum solutions become more effective in this experimental set-up.

Acknowledgments

Pedro Toledo is supported by a FPU fellowship from the Spanish Ministry of Education and Science.

References

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. *Eleventh International Conference on Data Engineering* (pp. 3–14). Taipei, Taiwan: IEEE Computer Society Press.

Friedler, F., Tarjan, K., Huang, Y. W., & Fan, L. T. Graph-theoretic approach to process synthesis: Axioms and theorems. *Chemical Engineering.*

Greco, G., Guzzo, A., & Manco, G. (2005). Mining and reasoning on workflows. *IEEE Transactions on Knowledge and Data Engineering*, *17*, 519–534. Senior Member-Domenico Sacca.

Gyapay, S., & Pataricza, A. (2003). A combination of petri nets and process network synthesis. *IEEE International Conference on Systems, Man and Cybernetics, Invited Sessions/Track on 'Petri Nets and Discrete Event Systems'*, 1167–1174.

Resende, M., & Ribeiro, C. (2005). Grasp with path-relinking: Recent advances and applications.

Rosen, K., Michaels, J., Gross, J., Grossman, J., & Shier, D. (2000). *Handbook of discrete and combinatorial mathematics.* CRC Press.

Rozinat, A., Alves de Medeiros, A., Gunther, C., Weijters, A., & van der Aalst, W. (2007). Towards an evaluation framework for process mining algorithms. *BPM Center Report*, *BPM-07-06.*

Posters

Learning to Plan and Planning to Learn via Merging Relational Machine Learning with Constraint Satisfaction (extended abstract)

Filip Železný and Onřej Kuželka {ZELEZNY,KUZELO1}FEL.CVUT.CZ
Czech Technical University in Prague, Czech Republic

The purpose of this submission is to present our newly started research project entitled "LeCoS: Merging Machine Learning with Constraint Satisfaction" concerned with exploring how to exploit relational machine learning (primarily inductive logic programming, ILP) to solve constraint satisfaction problems (primarily planning tasks), and vice versa. With this submission we wish to launch a discussion at the workshop about challenges and possible pitfalls of the previewed research directions of project as described below, and about further possible directions that we had not been aware of. The full paper submission following this abstract will also describe the lessons learned in our participation in the learning track of this year's international planning competition, where –in the frame of the mentioned project– we are developing ILP algorithms that learn relational heuristics from instances, solutions and process traces of complex planning problems.

The first main objective of the project is to exploit constraint satisfaction techniques in relational machine learning (RML). Most RML algorithms are based on methods of inductive logic programming (ILP), whose most typical goal is to induce a first-order logic theory explaining a set of classified examples (ground facts) using background knowledge, which also is a first-order logic theory. ILP has demonstrated numerous prominent applications (such as in bioinformatics), but the large computational demands of traditional ILP algorithms are generally considered an obstacle to its wider use. Although their worst-case run-times are often curtailed by suitable heuristics, these are usually domain-specific and do not translate across learning tasks. At the same time, the primary sources of ILP's complexity (the task of finding a consistent theory and the task of testing the consistency of a theory) are essentially CS problems (as shown previously by (Maloberti & Sebag, 2004)), although they are traditionally not viewed as such. Upon their suitable reformulation into the CS framework, relevant CS algorithms with their general and often powerful heuristics would naturally lend themselves to significantly alleviate the routine ILP tasks. There is indeed a wide scope for exploration, ranging from rather traditional CS methods to unorthodox techniques such as rapid random restarts which we have already shown to be highly beneficial in the CS task of theta-subsumption (Kuželka & Železný, 2009) conducted repeatedly in ILP systems.

The following set of proposed research directions is concerned with incorporating CS algorithms into machine learning (specifically inductive logic programming) algorithms, namely for the essential tasks of subsumption check and theory search.

- **CSP for subsumption checking.** The subsumption check procedure is at heart of practically every ILP system, used as an approximation to the generally undecidable problem of logical implication. The subsumption check is routinely used for scoring a candidate hypothesis (Horn rule) through verifying how many learning examples it logically entails. It is an NP-complete problem and procedures for its conversion into a CSP problem are generally known. However, only rare research works have addressed the exploitation of CSP algorithms with their native heuristics on such converted problems. Our aim is to explore an extended range of CSP techniques applied in this context and also investigate their impact for a more systemized set of particular settings under which the subsumption check is performed in ILP. Such settings correspond to various language-biases introduced previously in ILP research to constrain the often overly expressive and inefficiently manageable language of full Horn clause logic. Examples are determinate clauses or the relational feature language used in (Železný & Lavrač, 2006).
- **CSP for theory search.** The main challenge

here will be to reformulate the general ILP problem (finding a hypothesis that is complete and consistent with respect to learning examples) into a CSP problem. Clearly, the general ILP problem (where a hypothesis is a logic program) is undecidable, but ILP algorithms in practice always use some kind of a language bias (constraining the form of logic clauses as mentioned in the above item) turning the task into a different (decidable) complexity class. We will investigate what language biases makes the ILP problem NP-complete and feasibly convertible into a CSP problem. Then, using such biases, we will explore how CSP algorithms are effective at this alternative (w.r.t the state of the art) approach to solving the ILP problem.

- **Integration.** This objective aims at developing an algorithm combining the results of the previous two items. An expected outcome is thus an unorthodox ILP system in the form of a CSP procedure (subsumption check) embedded in another CSP procedure (theory search).

The second main objective is to exploit relational machine learning for efficient solving of constraint satisfaction problems, primarily complex planning tasks. There are indeed several genuine opportunities for ML to speed up CS. First, note that in a typical deployment of CS algorithms, CS problem instances are not generated randomly but are rather reformulations of a constrained class of tasks from a distribution pertaining to a certain domain. This can be exemplified by a repeated planning task (e.g. timetables for successive academic years in one institution) with only partially changing specifications (e.g. resources). Thus, successive CS instances may exhibit stable structural patterns, although these may be disguised to the naked eye. Here, machine learning algorithms have the potential to discover such patterns, occurring either in the instance specifications or in their solutions. In the former case, the patterns' occurrence may correlate with important quantities unknown prior to solving the task (e.g. the run-time) and thus may be used as predictors thereof. In the latter case, frequent patterns found in solutions of small (tractable) instances may straightforwardly be used as heuristics for solving larger instances of the same problem class: the CS algorithm would here prioritize the exploration of those candidate solutions which contain the 'promising' patterns. CS, viewed as an application domain for machine learning, is highly special in two respects. Firstly, the objects being learned about cannot be represented by a tuple of attribute values since they are relational structures. This implies that the advanced ML family - relational machine learning (such as ILP) - must be mobilized in such an application. Secondly, the fact that CS instances and - for tractable instances - even solutions, can be automatically generated, implies that a practically infinite supply of training examples would be available to the learning algorithm – a circumstance unheard of in usual ML applications. Both these aspects indicate an excellent opportunity for CS to become a killer application domain for relational machine learning and ILP.

This set of research directions will define the framework for learning from descriptions of previously solved CSP/Planning instances, their solutions as well as the computational processes employed in their solving, in order to augment subsequent solution processes.

- **Learning from CSP/Planning instances.** The main research questions will pertain to: (i) how CSP instances should effectively be represented as inputs to ML algorithms, mainly what descriptors (numeric, symbolic, relational, even including the position with respect to the phase-transition region of the instance solubility) should be used as descriptive features, (ii) what variables can and should be used as the target variables for supervised learning. Expected outcomes of this research are, for example, effectively invocable predictive models of (a) the actual time-complexity of a particular CSP instance (b) the most competent CSP algorithm, heuristic, etc. given the description of a particular instance.
- **Learning from CSP/Planning solutions.** Here we will primarily investigate how ML algorithms should be applied to learning from existing CSP/Planning solutions in order to discover frequent structural patterns therein. Following experimental work, we will try to trace the theoretical underpinnings which give rise to such patterns. Expected outcomes here lie in the automatic generation of CSP/Planning heuristics: discovered structural patterns will be reused as heuristics to streamline subsequent searches. A great expectation is in learning such heuristics from solutions of tractable CSP/Planning instances and in turn implant them into solving much harder instances, including those unsolvable by current CSP/Planning algorithms due to computational demands.
- **Learning from CSP/Planning instances and solutions.** This objective integrates the previous two. The aim is to connect the two sources

of information (instance descriptions and solutions). The main rationale is that the manifestation of certain frequent structural patterns in CSP/Planning solutions may not be implied by the choice of a general CSP/Planning subclass, but rather may be determined by certain properties of the specific instance in question. For example, the occurrence of certain patterns in instance descriptions may correlate with the occurrence of other patterns in the corresponding solutions. Such relationships can potentially be discovered by machine learning algorithms and used in a spirit similar to the previous item.

- **Learning from CSP/Planning processes.** While the previous objectives viewed CSP/Planning instances as learning examples, here we will proceed in a more fine-grained manner, by identifying objects arising in the course of CSP/Planning computation as learning examples. For instance, a learning example here may be a particular state in the CSP/Planning state-space exploration process. Expected benefits of such an approach include the potential ability e.g. to learn a description of states at which pruning is beneficial or where further exploration is especially promising, or to gradually learn the optimal value of search cutoff in a randomized restarted search strategy. Thus, one learning episode in this setting coincides with the entire course of solving one CSP/Planning instance; the predictive models, gradually tuned as the search proceeds, will be deployed to co-guide the search once their accuracy exceeds a suitable threshold. A significant supporting factor for this on-the-fly learning strategy is the enormous number of examples at hand: current CSP/Planning algorithms explore spaces extending to over millions of search nodes, thus–in lay language–there will likely be enough 'time' to train the model before its actual deployment in the given search process.

- **Learning from CSP/Planning instances and processes.** Here we aim to combine the CSP/Planning-instance-level learning with the process-level learning. An obvious kind of predictive model potentially discoverable in such a configuration is, for example, 'if the instance contains pattern P, and the current search node exhibits property R, prune all descendants of the current node.

References

Kuželka, O., & Železný, F. (2009). A restarted strategy for efficient subsumption testing. *Fundamenta Informaticae, spec. issue on multi-relational data mining. (Accepted).*

Maloberti, J., & Sebag, M. (2004). Fast theta-subsumption with constraint satisfaction algorithms. *Machine Learning*, *55*, 137–174.

F. Železný & N. Lavrač. (2006). Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, *62(1-2)*, 33–63.

Author Index