Selecting Machine Learning Algorithms Using the Ranking Meta-Learning Approach

Ricardo B. C. Prudêncio, Marcilio C. P. de Souto, and Teresa B. Ludermir

Center of Informatics, Federal University of Pernambuco,
Cidade Universitária - CEP 50732-970 - Recife (PE) - Brazil
{rbcp, mcps, tbl}@cin.ufpe.br

Abstract. In this work, we present the use of Ranking Meta-Learning approaches to ranking and selecting algorithms for problems of time series forecasting and clustering of gene expression data. Given a problem (forecasting or clustering), the Meta-Learning approach provides a ranking of the candidate algorithms, according to the characteristics of the problem's dataset. The best ranked algorithm can be returned as the selected one. In order to evaluate the Ranking Meta-Learning proposal, prototypes were implemented to rank artificial neural networks models for forecasting financial and economic time series and to rank clustering algorithms in the context of cancer gene expression microarray datasets. The case studies regard experiments to measure the correlation between the suggested rankings of algorithms and the ideal rankings. The results revealed that Meta-Learning was able to suggest more adequate rankings in both domains of application considered.

1 Introduction

One of the major challenges in many domains of Computational Intelligence, Machine Learning, Data Analysis and other fields is to investigate the capabilities and limitations of the existing algorithms in order to identify when one algorithm is more adequate than another to solve particular problems [1]. Traditional approaches to selecting algorithms involve, in general, costly trial-and-error procedures, or require expert knowledge, which is not always easy to acquire [2]. Meta-Learning for algorithm selection arises in this context as an effective solution, capable of automatically predicting algorithm's performance, thus assisting users in the choice of the most adequate techniques for dealing with the problems at hand [2–6].

In Meta-Learning, each *meta-example* is related to a learning problem and stores: (1) the features describing the problem, called *meta-features*; and (2) the *performance information* about one or more algorithms when applied to the problem. By receiving a set of such meta-examples, another learning system (the *meta-learner*) is applied to acquire knowledge relating the performance of the candidate algorithms and the descriptive features of the problems. The acquired knowledge can then be used to predict algorithm performance for new problems not seen during the Meta-Learning process and to recommend algorithms.

Different authors in Meta-Learning have developed techniques to suggest either one single algorithms or a small group of algorithms among the set of candidate ones. A more informative and flexible solution for algorithm selection is to provide a ranking of the candidate algorithms, since alternative algorithms can be eventually chosen by the users according to particular interests [7]. Ranking Meta-Learning approaches have been investigated in different case studies, mainly focused on classification and regression problems [1, 6, 7].

In this chapter, based on our previous works, we present the use of Meta-Learning for ranking algorithms in two different classes of problems: time series forecasting and clustering of gene expression data. Both domains are characterized by the existence of a variety of algorithms to be applied and a lack of useful guidelines to support algorithm selection. The Meta-Learning techniques originally proposed for classification and regression problems were extrapolated in our previous work to rank time series models and clustering techniques. The application of Meta-Learning in these domains was not deeply investigated yet. In a first case study, the Zoomed-Ranking approach was used to rank Artificial Neural Network models for forecasting financial and economic time series [8]. In a second case study, a Meta-Regression approach was used to rank clustering techniques for cancer gene expression [9].

The remaining of this paper is organized as follows. Section 2 presents a brief introduction on the topic of Meta-Learning. Section 3 presents the general architecture of our solution as well as some implementation issues. Section 4 brings a case study (implementation and experiments) performed in the domain of time series forecasting, followed by section 5 which presents a case study in the domain of clustering gene expression. Finally, section 6 concludes the paper with some final considerations.

2 Meta-Learning

There are different interpretations of the term *Meta-Learning* in the literature [3, 6, 10]. In our work, we focused on the definition of Meta-Learning as the automatic process of acquiring knowledge that relates the performance of learning algorithms to the features of the learning problems [2]. The acquired knowledge supports the task of algorithm selection, which has shown to be a difficult problem in many contexts [11].

The knowledge in Meta-Learning is commonly represented considering meta-features which describe learning problems. The meta-features are, in general, statistics describing the training dataset of the problem, such as number of training examples, number of attributes, correlation between attributes, class entropy, among others [7, 12, 1]. An alternative strategy to define meta-features is the *Landmarking* proposal [13]. This approach tries to relate the performance of the candidate algorithms to the performance obtained by simpler and faster designed learners, called *landmarkers*. Landmarking claims that some widely used meta-features are very time consuming, and hence, landmarking would be an economic approach to the characterization of problems and to provide useful information for the Meta-Learning process.

Regarding the performance information (the *target* in the Meta-Learning task), each meta-example may store a class attribute which indicates the best algorithm for the problem, among a set of candidates [14–18]. In this strict formulation of Meta-Learning, the class label for each meta-example is defined by performing a cross-validation experiment using the available dataset. The meta-learner is simply a classifier which predicts the best algorithm based on the meta-features of the problem.

In [19], the authors used an alternative approach to defining the performance information and hence to labeling meta-examples. Initially, 20 algorithms were evaluated through cross-validation on 22 classification problems. For each algorithm, the authors generated a set of meta-examples, each one associated either to the class label *applicable* or to the class label *non-applicable*. The class label *applicable* was assigned when the classification error obtained by the algorithm fell within a pre-defined confidence interval, and *non-applicable* was assigned otherwise. Each problem was described by a set of 16 meta-features and, finally, a decision tree was induced to predict the applicability of the candidate algorithms.

In [1], the authors performed the labeling of meta-examples by deploying a clustering algorithm. Initially, the error rates of 10 algorithms were estimated for 80 classification problems. From this evaluation, they generated a matrix of dimension 80 X 10, in which each row stored the ranks obtained by the algorithms in a single problem. The matrix was given as input to a clustering algorithm, aiming to identify groups (clusters) of problems in which the algorithms obtained specific patterns of performance (e.g. a cluster in which certain algorithms achieved a considerable advantage relative to the others). The meta-examples were then associated to the class labels corresponding to the identified clusters. Hence, instead of only predicting the best algorithm or the applicability of algorithms, the meta-learner can predict more complex patterns of relative performance.

Different approaches have been proposed in order to add new functionalities in the Meta-Learning process, especially to provide *rankings* of algorithms instead of recommending a single one. In [20, 21], for instance, a combination of strict meta-learners is used to recommend rankings of algorithms. In this approach, a strict meta-learner is built for each different pair (X, Y) of algorithms. Given a new learning problem, the outputs of the meta-learners are collected and then, points are credited to the algorithms according to the outputs. For instance, if 'X' is the output of meta-learner (X, Y) then the algorithm X is credited with one point. The ranking of algorithms is recommended for the new problem directly from the number of points assigned to the algorithms.

The Meta-Regression approach [22, 23] tries to directly predict the accuracy (or alternatively the error) of each candidate algorithm. The meta-learner in this case may be used either to select the algorithm with the highest predicted accuracy or to provide a ranking of algorithms based on the order of predicted accuracies. In [22], for instance, the authors obtained good results when a linear regression model was used to predict the accuracy of 8 different classification algorithms.

In the Zoomed-Ranking approach [24], the authors proposed to use instance-based learning in order to produce rankings of algorithms taking into account accuracy and execution time. In this approach, each meta-example stores the meta-features describing a learning problem, as well as the accuracy and execution time obtained by each candidate algorithm in the problem. Given a new learning problem, the Zoomed-Ranking retrieves the most similar past problem based on the similarity of meta-features. The ranking of algorithms is then recommended for the new problem by deploying a multi-criteria measure that aggregates the total accuracy and execution time obtained by the algorithms in the similar problems. More recently, the authors provided a deeper investigation of these ideas [7].

The concepts and techniques of meta-learning were mainly evaluated to select the best algorithms for classification and regression problems. In recent years, Meta-Learning has been extrapolated to other domains of application, such as in the selection of time series forecasting models [18], design of planning systems [25], combinatorial optimization [26], software engineering [27] and bioinformatics [9, 28, 29]. In such domains, Meta-Learning can be seen as tool for analysis of experiments performed by using a number of algorithms on a large set of problems that can be solved by these algorithms. The knowledge acquired from this analysis can be used to select algorithms for new problems. As highlighted in [5], Meta-Learning can be useful to a potentially large number of fields, since its developments can be extrapolated to learn about the behavior of algorithms on different classes of problems.

3 System Architecture and Implementation Issues

In this paper we present the use of *Ranking Meta-Learning* approaches in two different domains: time series forecasting and clustering of gene expression data [8, 9]. For each domain, we implemented a specific prototype in order to perform experiments and evaluate the usefulness of the Meta-Learning solution. In this section, we introduce a general architecture of Meta-Learning systems, as well as some implementation issues that guided us in the construction of the prototypes.

3.1 General Architecture

Figure 1 shows the general architecture of systems employed for, given a dataset, ranking the candidates algorithms. As it is common in Machine Learning, the system has two phases: training and use. In the training phase, the Meta-Learner (ML) extracts knowledge from the set of meta-examples stored in the Database (DB). Such a knowledge relates characteristics of the data to the performance of the candidate algorithms. In the case studies presented, these algorithms are either time series models or clustering techniques.

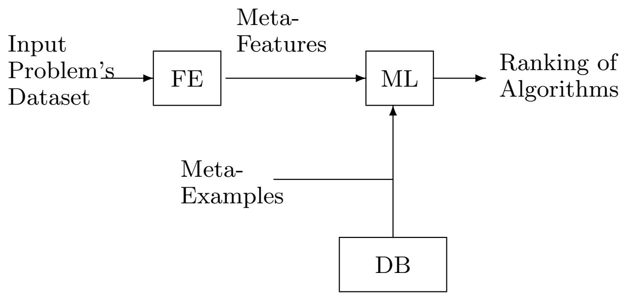

Fig. 1. System's architecture.

In the phase of use, given a new dataset, the Feature Extractor (FE) produces the values of the meta-features that describe these data. According to such values, the Meta-Learner (ML) module outputs a ranking of the available candidate algorithms. In order to do so, the ML module uses the knowledge previously provided as a result of the training phase.

The DB stores descriptions of datasets (i.e., meta-examples) used in the training phase. This set of meta-examples is semi-automatically created: (1) the choice of datasets and algorithms to be considered is a manual task; (2) the generation of the meta-features is automatically accomplished by the FE module; and (3) the performance of the candidate algorithms in each dataset is empirically obtained by directly applying each algorithm to the data and assessing the result yielded.

The ML module implements the chosen meta-learning approach to extracting knowledge (training phase) to be utilized in the choice or ranking of the candidate algorithms (use phase). As seen in Section 2, the Meta-Learning approaches implement one or more machine learning algorithms to execute such tasks. In this context, one could employ a learning technique to recommend one single algorithm from the set of candidate ones. Although this is a worthwhile approach, a more informative and flexible solution for algorithm selection is to output a ranking of the candidate algorithms to each dataset under analysis [7]. In this context, if enough resources are available, more than one algorithm could be employed with the data. Furthermore, if the one has some preference for a given subset of candidate algorithms, one can choose the algorithms that presented the best rank among the algorithms of interest.

3.2 Implementation Issues

To implement a system according to the architecture introduced in the previous section, one has to consider some important questions. Since the kind of dataset to be considered will have an impact on all the other aspects in the system's implementation, this is the first question to be addressed. Datasets are often collected from benchmarking repositories (e.g., the UCI repository in the case of classification and regression tasks) or artificially generated as performed in [30].

Then, one needs to specify which algorithms will be considered to form the set of candidate algorithms. The candidate algorithms should be selected in such a way to provide a wide range of characteristics, as well as to give some generality to the results.

The third question to be approached is which features will be employed by the FE module to describe the datasets. This decision depends on the kind of dataset being analyzed. For instance, in the context of classification problems, one can find standard sets of meta-features that have been used in the meta-learning field. This is the case of the *Data Characterization Tool*, developed within the METAL project[1]. In contrast, for time series forecasting and cluster analysis, since the application of meta-learning to these domains is relatively new, there is no such standard set of attributes. Nevertheless, one can follow some general guidelines to define them. For example, one should choose meta-features that can be reliably identified, preventing subjective analysis, such as visual inspection of plots. Subjective feature extraction is time consuming, requires expertise, and has a low degree of reliability [31]. One should also employ a manageable number of features in order to avoid a time consuming selection process.

4 Ranking Models for Time Series Forecasting

Time series forecasting has been used in several real world problems in order to eliminate losses resultant from uncertainty, as well as to support the decision-making process [32]. Several models can be used to forecast a time series. Selecting the most adequate model for a given time series, from a set of available models, may be a difficult task depending on the candidate models and the time series characteristics.

A straightforward solution to model selection is to perform an empirical evaluation (e.g. hold-out, cross-validation,...) using the available time series data, and compare the estimated performance obtained by the candidate models [33]. Despite its simplicity, this solution is costly for a large amount of series to forecast or several candidate models to evaluate [34].

A more efficient approach to selecting models is based on the development of expert systems [31], in which rules are designed to relate time series features (e.g. length, basic trend, autocorrelations...) and the candidate models performance. A landmarking work in this approach is the Rule-Based Forecasting system [31], in which an expert system with 99 rules was used to weight four forecasting methods. In the experiments performed using the expert system, the improvement in accuracy has shown to be significant. The main limitation of the expert system approach, however, is the difficulty in acquiring knowledge, since that good experts in time series forecasting are expensive and not always available [35]. This limitation may be even more drastic in the case of more complex models.

In order to minimize the above difficulty, in [18], the authors developed an original work which treats the model selection problem via Meta-Learning ap-

[1] http://www.cs.bris.ac.uk/~cgc/METAL

proaches. This solution is able not only to select the best model to forecast a given series, but also to provide more informative results, such as a ranking of the candidate models according to their performance in forecasting the given series. The viability of using Meta-Learning in the context of time series forecasting was confirmed in a number of different experiments [18, 17, 8, 36].

In this section, we reviewed the use of a specific Meta-Learning approach, the Zoomed-Ranking (ZR), to rank Artificial Neural Networks (ANNs) forecasting models [8]. The motivation was that, although ANNs represent a powerful approach to forecasting, there is not much knowledge to guide its usage, compared to the existing knowledge that supports the use of simpler linear models [34]. Hence, the investigation of ZR for ANN model selection contributes both to the research on Meta-Learning and to research on ANNs for time series forecasting.

In order to verify the viability of Meta-Learning, a prototype was implemented following the general architecture presented in section 3. The implemented prototype was used to select the following ANN models:

1. TDNN (Time Delay Neural Network) [37]: it corresponds to a feedforward network with time delays in the connections. The input layer receives a fixed time window of the series at hand (i.e. a fixed number of past values of the series), in order to forecast future values of the series. In our work, the time window size was defined by verifying the number of statistical significant autocorrelations in the series up to the limit of 3 past values. The number of hidden neurons was defined by deploying an out-of-sample experiment [33]. In this experiment, we evaluated the TDNN with 1, 2 and 3 hidden neurons on a series sample left out and depicted the best on in terms of forecasting accuracy. The network weights were trained using the Levenberg-Marquardt algorithm [38];
2. Time-Lagged RBF (Radial Basis Function) [39]: it corresponds to a traditional RBF neural network in which the input layer receives a time window of the series at hand (as in the TDNN model). The methodology adopted to define the time window size and the number of hidden neurons was the same one adopted for the TDNN;
3. SOMTAD (SOM with Temporal Activity Diffusion) [40]: it corresponds to a SOM network that creates temporally correlated neighborhoods in the output space. We defined the input layer of the SOM as the past 3 values of the series. In the SOM training, a 10x10 bi-dimensional map was adopted with learning rate of 0.3.

The candidate ANN models were used to forecast benchmarking time series related to financial, micro and macro-economic domains, available in the Time Series Data Library (TSDL) repository[2]. In the following sections, we provide more details of the implemented prototype.

[2] http://www-personal.buseco.monash.edu.au/~hyndman/TSDL

4.1 Feature Extractor

In this case study, we used in the FE module 5 different features to describe the TSDL series:

1. Length of the time series (L): number of observations of the series;
2. Basic Trend (BT): slope of the linear regression model. As higher this feature value, higher is the global trend of the series;
3. Test of Turning Points (TP): Z_t is a turning point if $Z_{t-1} < Z_t > Z_{t+1}$ or $Z_{t-1} > Z_t < Z_{t+1}$. The presence of a very large number or a very small number of turning points indicates that the series is not generated by a purely random process;
4. Average Coefficient of Autocorrelation (AC): average of the first 5 autocorrelation coefficients. Large values of this feature suggest a strong correlation between adjacent points in the series;
5. Type of the time series ($TYPE$): it is represented by 3 categories indicating the series domain, *finances, micro-economy* and *macro-economy.*

The first four features are directly computed using the series data and TYPE in turn is an information provided by the TSDL repository.

4.2 Database

In the construction of the Database, meta-examples were generated from the empirical evaluation of the three candidate models on different time series forecasting problems. Each meta-example for the ZR is related to a time series and stores: (1) the descriptive features of the series (as defined in the FE module); and (2) the forecasting error and the execution time obtained by each candidate model, when used to forecast the series.

In this case study, the accuracy and execution time of each model were collected by performing a hold-out experiment using the available time series data. Initially, the time series data is divided in two parts: the fit period and the test period. The test period in our prototype corresponds to the last observations of the series and the fit period corresponds to the remaining data. The fit period is used to train the ANN models. The trained models are then used to generate its individual forecasts for the test period. Finally, each meta-example is then composed by: the time series features extracted for describing the fit period, the mean absolute forecasting error obtained by the models in the test period and the time execution recorded during the ANN models training.

The process described above was applied to 80 different time series, and hence, a set of 80 meta-examples was generated. We highlight that such set of meta-examples actually stores the experience obtained from empirically evaluating the ANN models to forecast a large number of different time series. A Meta-Learning approach (as the ZR) will be able to use this experience to recommend rankings of models for new series only based on the time series features, without the need of performing new empirical evaluations.

4.3 Meta-Learner

The Zoomed-Ranking (ZR) meta-learning approach was used in the Meta-Learner module in order to rank the three ANN models. This approach is composed by two distinct phases: the Zooming and the Ranking phases, described here. Given a new time series to forecast, in the Zooming phase, a number of m meta-examples are retrieved from the training set according to a distance function that measures the similarity between time series features. The distance function implemented in the prototype was the L_1-norm defined as:

$$dist(x, x_i) = \sum_{j=1}^{p} \frac{|x^j - x_i^j|}{max_l(x_l^j) - min_l(x_l^j)} \tag{1}$$

In this equation, x is the description of the input series to be forecasted , x_i is the description of the i-th series in the training set and p is the number of meta-features. We used in the implemented prototype the L_1-norm as originally proposed in the ZR approach [24].

In the Ranking phase, a ranking of models is suggested, by aggregating the forecasting error and execution time stored in the m retrieved meta-examples. This is performed by deploying the Adjust Ratio of Ratios (ARR) measure [7], as defined in the equation:

$$ARR_{k,k'}^{i} = \frac{\frac{S_i^{k'}}{S_i^{k}}}{1 + AccD * log(\frac{T_i^k}{T_i^{k'}})} \tag{2}$$

In the above equation, S_i^k and T_i^k are respectively the forecasting error and execution time obtained by the model k on series i. The metric $ARR_{k,k'}^{i}$ combines forecasting error and execution time, to measure the relative performance of the models k and k' in the series i. The parameter $AccD$ is defined by the user and represents the relative importance between forecasting accuracy and execution time. $AccD$ assumes values between 0 and 1. The lower is the $AccD$ parameter, the higher is the importance given to accuracy relative to execution time.

The ratio of forecasting errors $S_i^{k'}/S_i^k$ can be seen as a measure of advantage of the model k in relation to model k', that is, a measure of relative benefit of model k (the higher is $S_i^{k'}/S_i^k$, the lower is the forecasting error of model k relative to model k'). In turn, the ratio of execution times $T_i^k/T_i^{k'}$ can be seen as a measure of disadvantage of the model k in relation to model k', that is, as measure of relative cost of model k. The ARR measure uses the ratio between a benefit and a cost measure to compute the overall quality of the candidate model k related to k'.

An aspect that should be observed regarding the time ratio is the fact that this measure has a much wider range of possible values than the ratio of accuracy rate. Therefore, if simple ratios of time were used, it would dominate the ARR measure. In this way, the effect of this range could be diminished by using the log of time ratios. We highlight that the use of log of time ratios was also adopted in [24, 7].

Finally, the ranking of models suggested to the input series is generated by aggregating the ARR information across the m retrieved meta-examples and K candidate models, as follows:

$$ARR_k = \frac{\sum_{k' \neq k} \sqrt[m]{\prod_{i \in Zoom} ARR^i_{k,k'}}}{K-1} \tag{3}$$

In the above equation, the $Zoom$ set represents the m retrieved meta-examples. The geometric mean in ARR is computed across the retrieved meta-examples and then the arithmetic mean across the candidate models. The ranking is suggested directly from ARR_k (the higher is the ARR_k value, the higher is the rank of model k). The geometric mean was used in order to satisfy the following property: $ARR_{k,k'} = 1/ARR_{k',k}$.

4.4 Experiments and Results

In the performed experiments, we collected 80 time series from the TSDL repository. Hence, a set of 80 meta-examples were generated by applying the procedure described in the section 4.2. This set was divided into 60 meta-examples for training and 20 meta-examples for testing the ZR approach.

The experiments were performed for different values of: (1) *AccD* parameter (0, 0.2, 0.4 and 0.6), which controls the relative importance of accuracy and time; and (2) the parameter m (1, 3, 5, 7, 9, 11 and 13 neighbors), which defines the neighborhood size in the Zooming phase.

In order to evaluate the performance of ZR, we deployed the Spearman Ranking Correlation coefficient (SRC). Given a series i, the SRC coefficient measures the similarity between the recommended ranking of models and the ideal ranking (i.e. the correct ordering of models taking into account the ARR measure computed in the series). The SRC for a series i is computed as:

$$SRC_i = 1 - \frac{6 * \sum_{k=1}^{K} (rr_{k,i} - ir_{k,i})^2}{K^3 - K} \tag{4}$$

In the equation, $rr_{k,i}$ and $ir_{k,i}$ are respectively the rank of model k in the recommended ranking and the ideal ranking for the series i and K is the number of candidate models. SRC_i assumes values between -1 and 1. Values near to 1 indicate that the two rankings have many agreement positions and values near to -1 indicate disagreement between the rankings. In order to evaluate the rankings generated for the 20 series in the test set, we calculated the average of SRC across these series.

The ZR approach was compared to a default ranking method [18], in which the ranking is suggested by aggregating the performance information for all training meta-examples, instead of using only the most similar ones. Despite its simplicity, the default method has been used as a basis of comparison in different case studies in the literature of Meta-Learning [7, 18, 8].

Table 1 shows the average values of SRC across the test series, considering the ZR approach and the default ranking. As it can be seen, the rankings recommended by ZR were in average more correlated to the ideal rankings when compared to the default method. The SRC average values for the default ranking are near to zero, indicating neutrality related to the ideal rankings. In fact, the average performance of each candidate model was very similar across the 20 test series, and then there was no clear preference among the models by default. In this way, the default ranking had a quality which was similar to a random choice of models. The ZR in turn obtained SRC values from 0.45 to 0.70, for all different experimental settings, indicating positive correlation to the ideal rankings.

Table 1. Average SRC coefficient across the 20 series in the test set.

	Average SRC			
	AccD = 0.0	**AccD = 0.2**	**AccD = 0.4**	**AccD = 0.6**
m = 1	0.45	0.47	0.50	0.45
m = 3	0.47	0.50	0.52	0.50
m = 5	0.47	0.50	0.52	0.50
m = 7	0.47	0.50	0.52	0.50
m = 9	0.62	0.50	0.52	0.50
m = 11	0.67	0.70	0.67	0.50
m = 13	0.67	0.65	0.62	0.45
Default	0.02	0.05	0.07	0.05

5 Ranking Clustering Techniques for Gene Expression Data

As previously mentioned, Meta-Learning had been used mostly for ranking and selecting supervised learning algorithms. Motivated by this, we extended the use of Meta-Learning approaches for clustering algorithms. We developed our case study in the context of clustering algorithms applied to cancer gene expression data generated by microarray [9].

Cluster analysis of gene expression microarray data is of increasing interest in the field of functional genomics [41–43]. One of the main reasons for this is the need for molecular-based refinement of broadly defined biological classes, with implications in cancer diagnosis, prognosis and treatment. Although the selection of the clustering algorithm for the analysis of microarray datasets is a very important question, there are in the literature few guidelines or standard procedures on how these data should be analyzed [44, 45].

The selection of algorithms is basically driven by the familiarity of biological experts to the algorithm rather than the features of the algorithms themselves and of the data [44]. For instance, the broad utilization of hierarchical clustering techniques is mostly a consequence of its similarity to phylogenetic methods, which biologists are often used to. Hence, in this context, by employing a Meta-Learning approach, our aim was to provide a framework to support non-expert users in the algorithm selection task [9].

In this section, we present a case study originally proposed in [9] in which a Meta-Regression approach was used to rank seven different candidate clustering methods: single linkage (SL), complete linkage (CL), average linkage (AL), k-means (KM), mixture model clustering (M), spectral clustering (SP), and Shared Nearest Neighbors algorithm (SNN) [46–48]. As it will be seen, meta-examples were generated from the evaluation of these clustering methods on 32 microarray datasets of cancer gene expression. the next sections provide the details of implementation for this case study, as well as the performed experiments.

5.1 Feature Extractor

In this case study, we used a set of eight descriptive attributes (meta-features). Some of them were first proposed for the case of supervised learning tasks.

1. LgE: $\log_{10}$ of the number of examples. A raw indication of the available amount of training data.
2. LgREA: $\log_{10}$ of the ratio of the number of examples by the number of attributes. A rough indicator of the number of examples available to the number of attributes.
3. PMV: percentage of missing values. An indication of the quality of the data.
4. MN: multivariate normality, which is the proportion of T^2 [49](examples transformed via T^2) that are within 50% of a Chi-squared distribution (degree of freedom equals to the number of attributes describing the example). A rough indicator on the approximation of the data distribution to a normal distribution.
5. SK: skewness of the T^2 vector. Same as the previous item.
6. Chip: type of microarray technology used (either cDNA or Affymetrix).
7. PFA: percentage of the attributes that were kept after the application of the attribute selection filter.
8. PO: percentage of outliers. In this case, the value stands for the proportion of T^2 distant more than two standard deviations from the mean. Another indicator of the quality of the data.

5.2 Database

Meta-examples in this case study are related to cancer gene expression microarray datasets. Each meta-example has two parts: (1) the meta-features describing a gene expression dataset; and (2) a vector with the ranking of the clustering algorithms for that dataset. A meta-regressor will use a set of such meta-examples to predict the algorithms' ranks for new datasets.

In order to assign this ranking for a dataset, we executed each of the seven clustering algorithms with a given dataset to produce the respective partitions. The number of clusters was set to be equal to the true number of the classes in the data. The known class labels was not used in any way during the clustering. As in other works, the original class labels constitute the *gold standard* against which we evaluate the clustering results [41, 46, 50].

For all non-deterministic algorithms, we ran the algorithm 30 times and picked the best partition. More specifically, in terms of the index to assess the success of the algorithm in recovering the gold standard partition of the dataset and building the ranking, we employed the corrected Rand index (cR) [46, 50]. The maximum value of the cR is 1, indicating a perfect agreement between the partitions. A value near 0 corresponds to a partition agreement found by chance. A negative value indicates that the degree of similarity between the gold standard partition and the partition yielded by the clustering algorithm is inferior to the one found by chance.

The cluster evaluation we adopt is mainly aimed at assessing how good the investigated clustering method is at recovering known clusters from gene expression microarray data. Formally, let $U = \{u_1, \ldots, u_r, \ldots, u_R\}$ be the partition given by the clustering solution, and $V = \{v_1, \ldots, v_c, \ldots, v_C\}$ be the partition formed by an a priori information independent of partition U (the gold standard). The corrected Rand is defined as:

$$cR = \frac{\sum_i^R \sum_j^C \binom{n_{ij}}{2} - \binom{n}{2}^{-1} \sum_i^R \binom{n_{i\cdot}}{2} \sum_j^C \binom{n_{\cdot j}}{2}}{\frac{1}{2}[\sum_i^R \binom{n_{i\cdot}}{2} + \sum_j^C \binom{n_{\cdot j}}{2}] - \binom{n}{2}^{-1} \sum_i^R \binom{n_{i\cdot}}{2} \sum_j^C \binom{n_{\cdot j}}{2}}$$

where (1) n_{ij} represents the number of objects in clusters u_i and v_j; (2) $n_{i\cdot}$ indicates the number of objects in cluster u_i; (3) $n_{\cdot j}$ indicates the number of objects in cluster v_j; (4) n is the total number of objects; and (5) $\binom{a}{b}$ is the binomial coefficient $\frac{a!}{b!(a-b)!}$.

Based on the values of the cR, the ranking for the algorithms is generated as follows. The clustering algorithm that presents the highest cR come higher in the ranking (i.e., the ranking value is equal to 1). Algorithms that generate partition with the same cR receive the same ranking number, which is the mean of what they would have under ordinal rankings.

5.3 Meta-Learner

Our system generates a ranking of algorithms for each dataset given as input. In order to create a ranking of K candidates (clustering algorithms), we use K regressors, each one responsible for predicting the ranking of a specific algorithm for the dataset given as input.

For building the regressor associated to a given algorithm k, we adopt the following procedure. First, we defined a set of meta-examples. Each meta-example corresponded to a dataset, described by a set of meta-features, with one of them representing the desired output. The value of the meta-attribute representing the desired output is assigned according to the ranking of the algorithm among all the seven ones employed to cluster the dataset. Next, we applied a supervised learning algorithm to each of the K regressors, which will be responsible for associating a dataset to a ranking.

As previously mentioned, we took into account seven clustering algorithms: SL, AL, CL, KM, M, SP and SNN. This led to construction seven regressors, $R_1, \ldots, R_7$, associated to, respectively, SL, AL, CL, KM, M, SP and SNN. For example, suppose that the outputs of the seven regressors for a new dataset are, respectively, 7, 5, 6, 1, 2, 4 and 3. Such an output means that model SL is expected to be the worst model (it is the last one in the ranking), AL is fifth best model model, CL the fourth one, KM is supposed to be better than all the others, as it is placed as first one in the ranking.

In our implementation, we employed the regression Support Vector Machine (SVM) algorithm, implemented in LIBSVM: a library for support vector machines [51]. A reason for this choice is that, in our preliminary results, SVMs showed a better accuracy than models such as artificial neural networks and k-NN.

5.4 Experiments and Results

We describe here the experiments that we developed in order to evaluate the performance of our prototype. Thirty two microarray datasets[3] (see Table 2). They are a set of benchmark microarray data presented in [52]. These datasets present different values for characteristics such as type of microarray chip (second column), number of patterns (third column), number of classes (fourth column), distribution of patterns within the classes (fifth column), dimensionality (sixth column), and dimensionality after feature selection (last column).

In terms of the datasets, it is worthwhile to point out that microarray technology is in general available in two different platforms, cDNA and Affymetrix [41–43]. Measurements of Affymetrix arrays are estimates on the number of RNA copies found in the cell sample, whereas cDNA microarrays values are ratios of the number of copies in relation to a control cell sample.

In order to remove uninformative genes for the case of Affymetrix arrays, we applied the following procedure. For each gene j (attribute), we computed the mean m_j. But before doing so, in order to get rid of extreme values, we discarded the 10% largest and smallest values. Based on this mean, we transform every value x^*_{ij} of example i and attribute j to:

$$y_{ij} = \log_2(x^*_{ij}/m_j)$$

After the previous transformation, we chose for further analysis genes whose expression level differed by at least l-fold, in at least c samples, from their mean expression level across samples. With few exceptions, the parameters l and c

[3] http://algorithmics.molgen.mpg.de/Supplements/CompCancer/ are included in this analysis

Table 2. Dataset description

Dataset	**Chip**	n	**Nr Classes**	**Dist. Classes**	d	**Filtered** d
Alizadeh-V1	cDNA	42	2	21,21	4022	1095
Alizadeh-V2	cDNA	62	3	42,9,11	4022	2093
Armstrong-V1	Affy	72	2	24,48	12582	1081
Armstrong-V2	Affy	72	3	24,20,28	12582	2194
Bhattacharjee	Affy	203	5	139,17,6,21,20	12600	1543
Bittner	cDNA	38	2	19, 9	8067	2201
Bredel	cDNA	50	3	31,14,5	41472	1739
Chen	cDNA	180	2	104,76	22699	85
Chowdary	Affy	104	2	62,42	22283	182
Dyrskjot	Affy	40	3	9,20,11	7129	1203
Garber	cDNA	66	4	17,40,4,5	24192	4553
Golub-V1	Affy	72	2	47,25	7129	1877
Gordon	Affy	181	2	31,150	12533	1626
Khan	cDNA	83	4	29,11,18,25	6567	1069
Laiho	Affy	37	2	8,29	22883	2202
Lapoint-V1	cDNA	69	3	11,39,19	42640	1625
Lapoint-V2	cDNA	110	4	11,39,19,41	42640	2496
Liang	cDNA	37	3	28,6,3	24192	1411
Nutt-V1	Affy	50	4	14,7,14,15	12625	1377
Nutt-V2	Affy	28	2	14,14	12625	1070
Nutt-V3	Affy	22	2	7,15	12625	1152
Pomeroy-V1	Affy	34	2	25,9	7129	857
Pomeroy-V2	Affy	42	5	10,10,10,4,8	7129	1379
Ramaswamy	Affy	190	14	11,10,11,11,22,10,11 10,30,11,11,11,11,20	16063	1363
Risinger	cDNA	42	4	13,3,19,7	8872	1771
Shipp	Affy	77	2	58,19	7129	798
Singh	Affy	102	2	58,19	12600	339
Su	Affy	174	10	26,8,26,23,12,11,7,27,6,28	12533	1571
Tomlins-V1	cDNA	104	5	27,20,32,13,12	20000	2315
Tomlins-V2	cDNA	92	4	27,20,32,13	20000	1288
West	Affy	49	2	25,24	7129	1198
Yeoh-V1	Affy	248	2	43,205	12625	2526

were selected in such a way as to produce a filtered dataset with around at least 10% of the original number of genes (features). It is important to point out that the data transformed with the previous equation is only used in the filtering step.

A similar filter procedure was applied for the case of cDNA microarray, but without the need to transform the data. In the case of cDNA microarray datasets, whose attributes (genes) could present missing values, we discarded the ones with more than 10% of missing values. The attributes that are kept and still present missing values have the values replaced for the respective mean value of the attribute.

For a given dataset, in order to generate the ranking, we took into account the configuration that obtained the best corrected Rand (see the second and third paragraphs in Section 5.2). We ran the algorithms with Euclidean distance, Pearson correlation and Cosine, but always with the number of clusters equal to the real number of classes in the dataset.

We assessed the performance of the meta-learners using the leave-one-out procedure. At each step, 31 examples are employed as the training set, and the remaining example is used to test the SVMs created. This step is repeated 32 times, utilizing at each time a different test example. The quality of a suggested ranking for a given dataset is evaluated by employing the average SRC, as described in equation 4, to measure the similarity between the suggested and the ideal rankings.

The result of our approach was compared to a default ranking method, in which the average ranking is suggested for all datasets. In our case, the default ranking was: SL=6.41, AL=4.60, CL=3.84, KM=2.31, M=3.40, SP=3.07, SNN=4.36. In Table 3, we illustrate the mean and standard deviation for the Spearman coefficient for the rankings generated by our approach and for the default ranking.

Table 3. Mean of the Spearman coefficient

Method	**SRC**
Default	0.59 ± 0.37
Meta-Leaner	0.75 ± 0.21

As it can be seen, the rankings generated by our method were more correlated to the ideal ranking. In fact, according to a hypothesis test, at a significance level of 0.05, the mean of the correlation value found with our method was significantly higher than that obtained with the default ranking.

6 Conclusion

In this paper, we present the results of using Ranking Meta-Learning approaches in two different domains of application: time series forecasting and clustering of gene expression. In the performed experiments, we observed that the rankings suggested by Meta-Learning were similar to the ideal rankings of algorithms observed in the available problems.

We can point out specific contributions of our work in the fields of time series forecasting and clustering of gene expression, which are domains of particular interest of many researchers. Different improvements can be considered in future work, such as increasing the number of meta-features (including the use of landmarking), investigating the viability of using artificial datasets in order to generate a larger database of meta-examples and performing experiments with other Meta-Learning approaches.

Finally, we highlight that Meta-Learning brings opportunities to researchers in different fields by providing general techniques that can be extrapolated to other algorithm selection tasks. Although the use of Meta-Learning in different domains has been increasing in recent years, there is still a large number of contexts in which it has not yet been investigated.

Acknowledgments: The authors would like to thank CNPq, CAPES and FACEPE (Brazilian Agencies) for their financial support.

References

1. Kalousis, A., Gama, J., Hilario, M.: On data and algorithms - understanding inductive performance. Machine Learning **54**(3) (2004) 275–312
2. Giraud-Carrier, C., Vilalta, R., Brazdil, P.: Introduction to the special issue on meta-learning. Machine Learning **54**(3) (2004) 187–193
3. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Journal of Artificial Intelligence Review **18**(2) (2002) 77–95
4. Koepf, C.: Meta-Learning: Strategies, Implementations, and Evaluations for Algorithm Selection. Infix (2006)
5. Smith-Miles, K.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Computing Surveys **41**(1) (2008) 1–25
6. Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning: Applications to Data Mining. Cognitive Technologies. Springer (2009)
7. Brazdil, P., Soares, C., da Costa, J.: Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. Machine Learning **50**(3) (2003) 251–277
8. dos Santos, P., Ludermir, T.B., Prudêncio, R.B.C.: Selection of time series forecasting models based on performance information. In: 4th International Conference on Hybrid Intelligent Systems. (2004) 366–371
9. de Souto, M.C.P., Prudencio, R.B.C., Soares, R.G.F., Araujo, D.A.S., Costa, I.G., Ludermir, T.B., Schliep, A.: Ranking and selecting clustering algorithms using a meta-learning approach. In: Proceedings of the International Joint Conference on Neural Networks, IEEE Computer Society (2008)
10. Jankowski, N., Grabczewski, K.: Building meta-learning algorithms basing on search controlled by machine complexity. In: IJCNN. (2008) 3601–3608
11. Duch, W.: What is computational intelligence and where is it going? In Duch, W., Mandziuk, J., eds.: Challenges for Computational Intelligence. Springer Studies in Computational Intelligence. Volume 63. Springer (2007) 1–13

12. Engels, R., Theusinger, C.: Using a data metric for preprocessing advice for data mining applications. In Prade, H., ed.: Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98), John Wiley & Sons (1998) 430–434
13. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Meta-learning by landmarking various learning algorithms. In: Proceedings of the 17th International Conference on Machine Learning, ICML'2000, San Francisco, California, Morgan Kaufmann (2000) 743–750
14. Aha, D.: Generalizing from case studies: A case study. In: Proceedings of the 9th International Workshop on Machine Learning, Morgan Kaufmann (1992) 1–10
15. Kalousis, A., Hilario, M.: Representational issues in meta-learning. In: Proceedings of the 20th International Conferente on Machine Learning. (2003) 313–320
16. Leite, R., Brazdil, P.: Predicting relative performance of classifiers from samples. In: 22nd Inter. Conf. on Machine Learning. (2005)
17. Prudêncio, R.B.C., Ludermir, T.B., de Carvalho, F.A.T.: A modal symbolic classifier to select time series models. Pattern Recognition Letters **25**(8) (2004) 911–921
18. Prudêncio, R.B.C., Ludermir, T.B.: Meta-learning approaches to selecting time series models. Neurocomputing **61** (2004) 121–137
19. D. Michie, D.J.S., Taylor, C.C., eds.: Machine Learning, Neural and Statistical Classification. Ellis Horwood, New York (1994)
20. Kalousis, A., Theoharis, T.: Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. Intelligent Data Analysis **3**(5) (1999) 319–337
21. Kalousis, A., Hilario, M.: Feature selection for meta-learning. Lecture Notes in Computer Science **2035** (2001) 222–233
22. Bensusan, H., Alexandros, K.: Estimating the predictive accuracy of a classifier. In: 12th European Conf. on Machine Learning. (2001) 25–36
23. Koepf, C., Taylor, C.C., Keller, J.: Meta-analysis: Data characterisation for classification and regression on a meta-level. In: Proceedings of the International Symposium on Data Mining and Statistics. (2000)
24. Soares, C., Brazdil, P.: Zoomed ranking - selection of classification algorithms based on relevant performance information. Lecture Notes in Computer Science **1910** (2000) 126–135
25. Tsoumakas, G., Vrakas, D., Bassiliades, N., Vlahavas, I.: Lazy adaptive multicriteria planning. In: Proceedings of the 16th European Conference on Artificial Intelligence, ECAI04. (2004) 693–697
26. Smith-Miles, K.: Towards insightful algorithm selection for optimisation using meta-learning concepts. In: Proceedings of the IEEE International Joint Conference on Neural Networks 2008. (2008) 4118–4124
27. Caiuta, R., Pozo, A.: Selecting software reliability models with a neural network meta classifier. In: Proceedings of the Joint International Conference on Neural Networks. (2008)
28. Nascimento, A.C.A., Prudêncio, R.B.C., Souto, M.C.P., Costa, I.G.: Mining rules for the automatic selection process of clustering methods applied to cancer gene expression data. Lecture Notes in Computer Science **5769** (2009) 20–29
29. Souza, B., Soares, C., Carvalho, A.: Meta-learning approach to gene expression data classification. International Journal of Intelligent Computing and Cybernetics **2**(2) (2000) 285–303
30. Soares, C.: Uci++, improved support for algorithm selection using datasetoids. Lecture Notes in Computer Science **5476** (2009) 499–506

31. Adya, M., Collopy, F., Armstrong, J., Kennedy, M.: Automatic identification of time series features for rule-based forecasting. International Journal of Forecasting **17**(2) (2001) 143–157
32. Montgomery, D.C., Johnson, L.A., Gardiner, J.S.: Forecasting and Time Series Analysis. MacGraw Hill, New York (1990)
33. Tashman, L.J.: Out-of-sample tests of forecasting accuracy: An analysis and review. International Journal of Forecasting **16** (2000) 437–450
34. Prudêncio, R.B.C., Ludermir, T.B.: Selection of models for time series prediction via meta-learning. In: Proceedings of the Second International Conference on Hybrid Systems, IOS Press (2002) 74–83
35. Arinze, B.: Selecting appropriate forecasting models using rule induction. Omega-International Journal of Management Science **22**(6) (1994) 647–658
36. Prudêncio, R.B.C., Ludermir, T.B.: A machine learning approach to define weights for linear combination of forecasts. In: 16th International Conference on Artificial Neural Networks. (2006) 274–283
37. Lang, K.J., Hinton, G.E.: A time-delay neural network architecture for speech recognition. Technical Report CMU-DS-88-152, Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, PA (1988)
38. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Quarterly Journal of Applied Mathmatics **II**(2) (1944) 164–168
39. Haykin, S.: Neural Networks: A Comprehensive Foundation. Macmillan College Publishing Company, New York (1994)
40. Principe, J., Euliano, N., Garania, S.: Principles and networks for self-organization in space-time. Neural Networks **15** (2002) 1069–1083
41. Monti, S., Tamayo, P., Mesirov, J., Golub, T.: Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. Machine Learning **52** (2003) 91–118
42. Quackenbush, J.: Computational analysis of cDNA microarray data. Nature Reviews **6**(2) (2001) 418–428
43. Slonim, D.: From patterns to pathways: gene expression data analysis comes of age. Nature Genetics **32** (2002) 502–508
44. D'haeseleer, P.: How does gene expression clustering work? NATURE BIOTECHNOLOGY **23**(12) (2005) 1499–1501
45. de Souto, M.C., Costa, I.G., de Araujo, D.S., Ludermir, T.B., Schliep, A.: Clustering cancer gene expression data: a comparative study. BMC Bioinformatics **9** (2008) 497+
46. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice Hall. (1988)
47. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Transactions on Neural Networks **16**(3) (2005) 645–678
48. Ertoz, L., Steinbach, M., Kumar, V.: A new shared nearest neighbor clustering algorithm and its applications. In: Workshop on Clustering High Dimensional Data and its Applications. (2002) 105–115
49. Johnson, R.A., Wichern, D.W.: Applied Multivariate Statistical Analysis. Fifth edition edn. Prentice Hall (2002)
50. Milligan, G.W., Cooper, M.C.: A study of standardization of variables in cluster analysis. Journal of Classification **5** (1988) 181–204
51. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. (2001) Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.
52. de Souto, M.C.P., Costa, I.G., Araujo, D.S.A., Ludermir, T.B., Schliep, A.: Clustering cancer gene expression data - a comparative study. BMC Bioinformatics **9** (2008) 497–520