International Journal of Hybrid Intelligent Systems 5 (2008) 1–12
IOS Press

Selective generation of training examples in active meta-learning

Ricardo B.C. Prudêncio and Teresa B. Ludermir
Centro de Informática, Universidade Federal de Pernambuco, Caixa Postal 7851 – CEP 50732-970, Recife (PE), Brazil

Abstract. Meta-Learning has been successfully applied to acquire knowledge used to support the selection of learning algorithms. Each training example in Meta-Learning (i.e. each meta-example) is related to a learning problem and stores the experience obtained in the empirical evaluation of a set of candidate algorithms when applied to the problem. The generation of a good set of meta-examples can be a costly process depending for instance on the number of available learning problems and the complexity of the candidate algorithms. In this work, we proposed the Active Meta-Learning, in which Active Learning techniques are used to reduce the set of meta-examples by selecting only the most relevant problems for meta-example generation. In an implemented prototype, we evaluated the use of two different Active Learning techniques applied in two different Meta-Learning tasks. The performed experiments revealed a significant gain in the Meta-Learning performance when the active techniques were used to support the meta-example generation.

1. Introduction

In several domains of application, such as in Machine Learning, there is a diversity of algorithms that can be considered as candidates to solve particular problems. One of the most difficulty tasks in such domains is to predict when one algorithm is better than another to solve a problem at hand [13]. Traditional approaches to predicting the performance of algorithms involve, in general, costly trial-and-error procedures, or require expert knowledge, which is not always easy to acquire [9]. Meta-Learning arises in this context as an effective solution, capable of automatically predicting algorithms performance for a given problem, thus assisting users in the process of algorithm selection [9, 44].

Each training example for Meta-Learning (i.e., each *meta-example*) stores the experience obtained from applying a number of candidate algorithms to a particular problem in the past. More specifically, each meta-example stores: (1) the features used to describe the problem; and (2) information about the performance obtained by the algorithms in the problem. By receiving a set of such meta-examples as input, another learning algorithm (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 is then used to support the selection of adequate algorithms for new problems available in the future.

A difficulty that can be pointed out in the Meta-Learning process is related to the generation of meta-examples. In order to generate a meta-example from a given past problem, it is necessary to perform an empirical evaluation (e.g. cross-validation) to collect the performance information of the candidate algorithms. Hence, generating a whole set of meta-examples may be expensive, depending for instance on the number and complexity of the candidate algorithms, the efficiency of the procedure used for algorithm evaluation, the number of available problems and amount of data available in each problem.

In this paper, we present the proposal of *Active Meta-Learning* in which Active Learning techniques [7] are used to support the generation of meta-examples. The general motivation of Active Learning is to reduce the number of training examples, at same time maintaining the performance of the learning algorithms. In our specific proposal, Active Learning techniques are used to reduce the number of meta-examples by selecting only the most relevant problems for meta-example generation, and consequently, reducing the number of empirical evaluations performed on the candidate algorithms.

In order to evaluate the proposed solution, we implemented a prototype in which two different Active Learning methods were used to generate meta-examples for a k-NN (k-Nearest Neighbors) meta-learner. The prototype was evaluated in two different case studies, corresponding to meta-learning tasks proposed in previous work. The experiments performed in both case studies revealed a significant gain in the meta-learner performance when the Active Learning methods were used for generating meta-examples.

Section 2 brings a brief presentation of Meta-Learning, followed by Section 3 which presents some Active Learning techniques. Section 4 describes the proposed solution and the implemented prototype, followed by Section 5 which presents the performed experiments and obtained results. Finally, Section 6 concludes the paper.

2. Meta-Learning

According to [44], there are different interpretations of the term *Meta-Learning*. 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 [9]. In this context, each *meta-example* is related to a learning problem and stores: (1) the features describing the problem, called *meta-features*; and (2) information about the performance of one or more algorithms when applied to the problem. The *meta-learner* is a learning system that receives as input a set of such meta-examples and then acquires knowledge used to predict the algorithms performance for new problems being solved.

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 [5,8, 13]. In a strict formulation of Meta-Learning, each meta-example stores, as performance information, a class attribute which indicates the best algorithm for the problem, among a set of candidates [2,12,16,26, 27]. In this case, 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 [22], the authors used an alternative approach 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 [13], 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. In [10,11], for instance, a set of different meta-learners is used not only to predict a class label associated to algorithms performance, but also to recommend a ranking 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 [4,14] 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 [4], 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 [38], the authors originally proposed the use of 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 [5].

The Landmarking approach [24] tries to relate the performance of the candidate algorithms to the performance obtained by simpler and faster designed learners, called *landmarkers*. This approach claims that some widely used meta-features are very time consuming, and hence, landmarking would be an economic approach to the characterization of learning problems and to provide useful information for the Meta-Learning process.

The concepts and techniques of meta-learning were mainly evaluated to select the best algorithms for classification tasks. In recent years, Meta-Learning has been extrapolated to other domains of application, such as in the selection of time series forecasting models [27] and in the design of planning systems [42]. 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 the algorithms. The knowledge acquired from this analysis can be used to select algorithms for new problems. In this sense, Meta-Learning is a more general framework and it is expected to be also useful to algorithm selection in problems related to different domains of application.

3. Active Learning

In the traditional supervised learning, the learner is trained by taking as input a set of randomly chosen training examples. This approach is referred in the literature as *passive learning* [31], and it has been broadly applied in several applications. However, both empirical and theoretical studies have shown that the learning performance can be improved when the learner is allowed to *ask questions* to the teacher, i.e to decide which data will be included in the training set.

Active Learning is a paradigm of Machine Learning in which the learning algorithm has some control over the inputs on which it trains [7]. The main objective of this paradigm is to reduce the number of training examples, at same time maintaining (or even improving) the performance of the learning algorithm. Active Learning is ideal for learning domains in which the acquisition of labeled examples is a costly process, such as image recognition [20], text classification [30,41], speech recognition [31] and information filtering [34].

One of the earliest approaches to Active Learning is the *membership query* [3], in which the learner artificially creates informative examples in the input domain and asks the teacher to annotate it. According to [23], this approach is limited in practice since it is likely to produce examples that do not have any sense in the domain of application (e.g. an unrecognizable image).

According to [20], previous work in Active Learning has been mainly concentrated in the *selective sampling* approach. In this approach, the learning algorithm has access to a set of (natural) unlabeled examples and, at each moment, selects the most informative ones. The teacher is asked to label the selected examples, which will be then included in the training set. According to [23], the selective sampling methods can be distinguished on three main categories, *uncertainty-based* methods, *version space reduction* methods and *error reduction* methods, described just below.

In *uncertainty-based* methods [18,20,30,35] for selective sampling, in order to select unlabeled examples, the learner initially uses the currently labeled examples to generate a prediction for each unlabeled example. Following, a degree of uncertainty of the provided prediction is assigned for each unlabeled example. Finally, the active method selects the example with highest uncertainty. According to [23], these methods can be straightforwardly applied to many different learning algorithms. A limitation of uncertainty based methods, however, is that they often select examples that are outliers [32]. Such examples have in fact a high degree of uncertainty but they should not be considered as informative.

In the *version space reduction* methods (also called committee-based methods) [1,19,21,36,41], a subset of the version space (i.e. a committee of hypotheses consistent with the current labeled examples) is generated and then applied to make predictions for the unlabeled examples. The method then selects the unlabeled example on which the members of the committee most disagree. These methods are actually related to uncertainty methods, since the degree of disagreement on the committee's predictions can be viewed as a measure of uncertainty. Different committee-based methods were proposed in the literature. These methods can be mainly distinguished by the way of generating the committees, which includes, for instance, the use of bagging and boosting algorithms [1].

In the *error reduction* methods [20,32,40], the selected unlabeled example is the one that minimizes the expected error rate of the learner, once labeled and included in the training set. Since the true label of an unlabeled example is not known a priori, the expected error rate is an average rate over the possible labels that the example could be assigned to. Although these methods have obtained good performance compared to other selective sampling methods, they are computationally expensive, since for each candidate example and possible label, it is necessary to re-train the learner in order to compute the expected error rate [32].

As it was said, Active Learning methods have been applied to improve learning performance in different applications. As it will be seen, we propose here an original work that use of Active Learning methods in the context of Meta-Learning.

4. Active Meta-Learning

As seen in Section 2, in order to generate a meta-example from a given problem, it is necessary to perform an empirical evaluation of the candidate algorithms on the problem. The empirical evaluation is performed in order to collect the performance information of the algorithms, and hence, to define the target attribute in the Meta-Learning process (e.g. the class corresponding to the best algorithm).

Although the proposal of Meta-Learning is to perform the empirical evaluation of the algorithms only in a limited number of problems, the cost of generating a set of meta-examples may be high depending on a number of aspects, including, for instance, the methodology of empirical evaluation, the number of available problems, and the number and complexity of the candidate algorithms. In this context, the use of Active Learning may improve the Meta-Learning process by reducing the number of required meta-examples, and consequently the number of empirical evaluations on the candidate algorithms.

Figure 1 presents the architecture of system following our proposal, which has three phases. In the meta-example generation, the Active Learning (AL) module selects from a base of problems, those ones considered the most relevant for the Meta-Learning task. The selection of problems is performed based on a pre-defined criteria implemented in the module, which takes into account the features of the problems and the current knowledge of the Meta-Learner (ML). The candidate algorithms are then empirically evaluated on the selected problems, in order to collect the performance information related to the algorithms. Each generated meta-example (composed by meta-features and performance information) is then stored in an appropriate database.

In the training phase, the Meta-Learner acquires knowledge from the database of meta-examples generated by the AL module. This knowledge associates meta-features to the performance of the candidate algorithms. The acquired knowledge may be refined as more meta-examples are provided by the AL module.

In the use phase, given a new input problem, the Feature Extractor (FE) module extracts the values of the meta-features. According to these values, the ML module predicts the performance information of the algorithms. For that, it uses the knowledge previously acquired as a result of the training phase.

In [28,29], we presented the initial experiments performed to evaluate the viability of the proposed solution. In the current work, we present new experiments that evaluated the proposed solution. Here, we implemented a prototype which was applied in two different case studies. In this prototype, the k-Nearest Neighbors (k-NN) algorithm was used in the ML module, and two different Uncertainty-based Active Learning methods were used in the AL module. In the next sections, we provide more details of the proposed implemented prototype. In Section 5, we present the two case studies as well as the experiments and obtained results.

4.1. Meta-Learner

The Meta-Learner in the prototype corresponds to a conventional classifier, and it is applicable to tasks in which the performance information is formulated as a class attribute (e.g. the class associated to the best algorithm or the class related to patterns of performance). In the implemented prototype, we used the k-NN algorithm which has some advantages when applied to Meta-Learning [5]. For instance, when a new meta-example becomes available, it can be easily integrated without the need to initiate re-learning [5]. In this

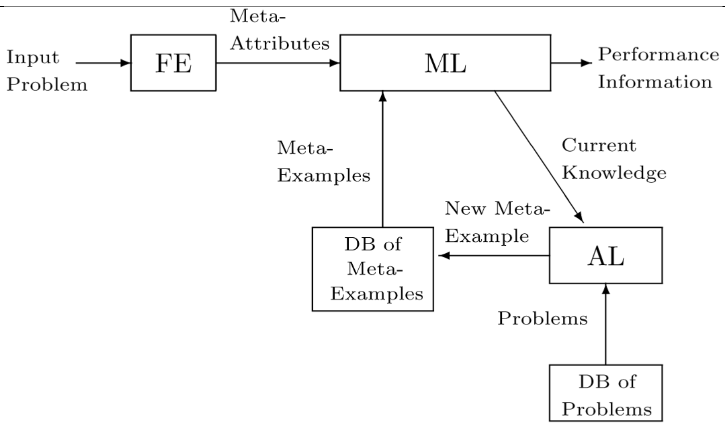

Fig. 1. System architecture.

section, we provide a description of the meta-learner based on k-NN. Other classes of learning algorithms, such as neural networks and support vector machines, can be applied in future work as meta-learners.

Let $E = \{e_1, \ldots, e_n\}$ be the set of n problems used to generate a set of n meta-examples $ME = \{me_1, \ldots, me_n\}$. Each meta-example is related to a problem and stores the values of p features $X_1, \ldots, X_p$ (implemented in the FE module) for the problem and the value of a class attribute C, which is the performance information

Let $\mathcal{C} = \{c_1, \ldots, c_L\}$ be the domain of the class attribute C, which has L possible class labels. In this way, each meta-example $me_i \in ME$ is represented as the pair $(\mathbf{x}_i, C(e_i))$ storing: (1) the description $\mathbf{x}_i$ of the problem e_i, where $\mathbf{x}_i = (x_i^1, \ldots, x_i^p)$ and $x_i^j = X_j(e_i)$; and (2) the class label associated to e_i, i.e. $C(e_i) = c_l$, where $c_l \in \mathcal{C}$.

Given a new input problem described by the vector $\mathbf{x} = (x^1, \ldots, x^p)$, the k-NN meta-learner retrieves the k most similar meta-examples from ME, according to the distance between meta-attributes. The distance function (*dist*) implemented in the prototype was the unweighted L_1-Norm, defined as:

$$\text{dist}\,(\mathbf{x}, \mathbf{x}_i) = \sum_{j=1}^{p} \frac{|x^j - x_i^j|}{\max_i(x_i^j) - \min_i(x_i^j)} \quad (1)$$

The prediction of the class label for the new problem is performed according to the number of occurrences (votes) of each $c_l \in \mathcal{C}$ in the class labels associated to the retrieved meta-examples.

4.2. Active Learning

The ML module acquires knowledge from a set of *labeled* meta-examples. The AL module receives a set of *unlabeled* meta-examples, associated to problems in which the candidate algorithms were not yet evaluated. The AL module incrementally selects unlabeled meta-examples to be used for generating new labeled meta-examples. In the prototype, the AL module considers two different uncertainty-based methods (see Section 3) which selects the unlabeled example for which the current learner has the highest uncertainty in its prediction. These methods are described as follows.

4.2.1. Active Learning: Uncertainty Method A

The classification uncertainty of the k-NN algorithm is defined in [20] as the ratio of: (1) the distance between the unlabeled example and its nearest labeled neighbor; and (2) the sum of the distances between the unlabeled example and its nearest labeled neighbors of different classes.

In the above definition, a high value of uncertainty indicates that the unlabeled example has nearest neighbors with similar distances but conflicting labeling. Hence, once the unlabeled example is labeled, it is expected that the uncertainty of classification in its neighborhood should be reduced.

In our context, let E be the set of problems associated to the labeled meta-examples, and let $\widetilde{E}$ be the set of problems used to generate unlabeled meta-examples. Let E_l be the subset of labeled problems associated to the class label c_l, i.e. $E_l = \{e_i \in E | C(e_i) = c_l\}$. Given the set E, the classification uncertainty of k-NN for each $\widetilde{e} \in \widetilde{E}$ is defined as:

$$\mathcal{S}(\widetilde{e}|E) = \frac{\min_{e_i \in E}\ \text{dist}\,(\widetilde{\mathbf{x}}, \mathbf{x}_i)}{\sum_{l=1}^{L} \min_{e_i \in E_l}\ \text{dist}\,(\widetilde{\mathbf{x}}, \mathbf{x}_i)} \quad (2)$$

In the above equation, $\widetilde{\mathbf{x}}$ is the description of problem $\widetilde{e}$. The AL module then selects, for generating a new labeled meta-example, the problem $\widetilde{e}^* \in \widetilde{E}$ with highest uncertainty:

$$\widetilde{e}^* = \text{argmax}_{\widetilde{e} \in \widetilde{E}} \mathcal{S}(\widetilde{e}|E) \tag{3}$$

Finally, the selected problem is labeled (i.e. the class value $C(\widetilde{e}^*)$ is defined), through the empirical evaluation of the candidate algorithms using the avaliable data of the problem.

4.2.2. *Active Learning: Uncertainty Method B*

In the second method considered in the AL module, we adopted the concept of entropy to define classification uncertainty. Assume that the k-NN can predict for each given example a probability distribution over the possible class values. Formally, the probability distribution for an unlabeled problem $\widetilde{e}$ can be represented as:

$$\begin{aligned} p_C(\widetilde{e}|E) = (p(C(\widetilde{e}) = c_1|E), \ldots, p(C(\widetilde{e}) \\ = c_L|E)) \end{aligned} \tag{4}$$

According to [39], the entropy of the probability distribution reflects the certainty of the classifier in the predicted class value. The entropy of the probability distribution is computed as:

$$\begin{aligned} \text{Entropy } (\widetilde{e}|E) &= -\sum_{l=1}^{L} p(C(\widetilde{e}) \\ &= c_l|E) * \log_2 p(C(\widetilde{e}) \\ &= c_l|E) \end{aligned} \tag{5}$$

If the probability distribution is highly spread, the value of entropy will be high, which indicates that the classifier is not certain in its prediction. On the other hand, if the distribution is highly focused on a single class label, the entropy is low, indicating a low degree of uncertainty in predicted class value.

As in the previous section, in this method, the AL module selects the problem $\widetilde{e}^* \in \widetilde{E}$ with highest uncertainty defined by the entropy measure:

$$\widetilde{e}^* = \text{argmax}_{\widetilde{e} \in \widetilde{E}} \text{ Entropy } (\widetilde{e}|E) \tag{6}$$

In our work, the class probability distribution for a given example is estimated by using the number of votes that each class label received among the retrieved meta-examples. In the next section, we present the case studies that evaluated the two implemented active methods.

5. Case studies

In this section, we present the application of the implemented prototype to two different case studies that correspond to two meta-learning tasks originally presented in previous work [25,28]. Each case study provides a set of meta-examples which was used in the current work to perform experiments to evaluate the implemented prototype.

5.1. *Case Study I*

In the first case study, the implemented prototype was evaluated in a meta-learning task originally proposed in [25] which consisted in selecting between two candidate algorithms for time series forecasting problems: the Time-Delay Neural Network (TDNN) [15] and the Simple Exponential Smoothing model (SES) [6]. In [25], a set of meta-examples was generated from the evaluation of TDNN and SES on 99 time series collected from the Time Series Data Library.[1] Hence, 99 meta-examples were generated.

Each meta-example was related to a single time series and stored: (1) the values of $p = 10$ meta-attributes (features describing the time series data) and (2) a class attribute which indicated the best forecasting model (SES or TDNN) for that series. The set of meta-attributes was composed by:

1. Length of the time series (X_1): number of observations of the series;
2. Mean of the absolute values of the 5 first-order autocorrelations (X_2): the autocorrelation coefficient r_s measures the correlation in a given series $Z_t(t = 1 \ldots T)$ at different points in time (see Eq. (7), where s is the order of the autocorrelation, T is the length of the series and $\overline{Z}$ is the mean value of the series). High values of this feature suggests that the value of the series at a time point is very dependent of the values in recent past points.

$$r_s = \frac{\sum_{t=1}^{T-s}(Z_t - \overline{Z})(Z_{t+s} - \overline{Z})}{\sum_{t=1}^{T}(Z_t - \overline{Z})} \tag{7}$$

3. Test of significant autocorrelations (X_3): presence of at least one significant autocorrelation taking into account the first 5;

[1]TSDL – http://www-personal.buseco.monash.edu.au/˜hyndman/TSDL

4. Significance of the first, second and third autocorrelation (X_4, X_5 and X_6): indicates significant dependences in more recent past points;
5. Coefficient of variation (X_7): measures the degree of instability in the series;
6. Absolute value of the skewness and kurtosis coefficient (X_8 and X_9): measure the degree of non-normality in the series;
7. Test of Turning Points for randomness (X_{10}): 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 or a very low number of turning points in a series suggests that the series is not generated by a purely random process.

The above features are classical measures for describing time series and can be quickly computed even for a large number of series [37]. Different works in the literature used at least part of these features for model selection purposes [26,27,37,43].

In this case study, the labeling of a time series (i.e. definition of the class attribute for training meta-examples) is performed through the empirical evaluation of TDNN and SES in forecasting the series. For this, a hold-out experiment was performed, as described in [25]. Given a time series, its data was divided into two parts: the fit period and the test period. The test period consists on the last 30 points of the time series and the fit period consists on the remaining data. The fit data was used to calibrate the parameters of both models TDNN and SES. Both calibrated models were used to generate one-step-ahead forecasts for the test data. Finally, the class attribute was assigned as the model which obtained the lowest mean absolute forecasting error on the test data.

5.1.1. Experiments

The prototype was evaluated for different configurations of the k-NN meta-learner (with $k = 1$, 3, 5, 7, 9 and 11 nearest neighbors). For each configuration, a leave-one-out experiment was performed to evaluate the performance of the meta-learner, also varying the number of meta-examples provided by the Active Learning module. This experiment is described just below.

At each step of leave-one-out, one problem is left out for testing the ML module, and the remaining 98 problems are considered as candidates to generate meta-examples. The AL module progressively includes one meta-example in the training set of the ML module, up to the total number of 98 training meta-examples. At each included meta-example, the ML module is judged on the test problem left out, receiving either 1 or 0 for failure or success. Hence, a curve with 98 binary judgments is produced for each test problem. Finally, the curve of error rates obtained by ML can be computed by averaging the curves of judgments over the 99 steps of the leave-one-out experiment.

The above procedure was applied for each Uncertainty method considered in the AL module (see Section 4.2). As a basis of comparison, the same above experiment was applied to each configuration of k-NN, but using in the AL module a Random Sampling method for selecting unlabeled problems. According to [20], despite its simplicity, the random method has the advantage of performing a uniform exploration of the example space.

Finally, we highlight that the experiments were performed in 30 different runs for each configuration of the k-NN meta-learner.

5.1.2. Results

Figure 2 presents the curves of error rates obtained by the k-NN meta-learner averaged across the different configurations of the parameter k and runs of experiments. The figure presents the average curve obtained when the three methods were used: the Uncertainty Method A (which was proposed in [20]), the Uncertainty Method B (which used the concept of entropy) and the Random Sampling method.

As it is expected, for all methods, the error rate obtained by the ML module decreased as the number of meta-examples in the training set increased. However, the error rates obtained by deploying the Uncertainty methods were, in general, lower than the error rates obtained by deploying the Random method. In absolute terms, the Uncertainty Method A steadily achieved better performance compared to the Random method from 13 to 87 meta-examples included in the training set. The Uncertainty Method B, in turn, obtained better performance compared to the Random method from 13 to 95 points in the curve or error rates (excepting the point 46). Hence, for both Uncertainty methods, an absolute gain in performance was obtained in the most part of the curve of error rates.

In the performed experiments, we also compared the Uncertainty methods to the Random Sampling in statistical terms, by applying a t-test (95% of confidence) to the difference of error rates. Figure 3 presents the points in the curve of error rates in which the Uncertainty methods were statistically better than the Random method. The Uncertainty Method A obtained a

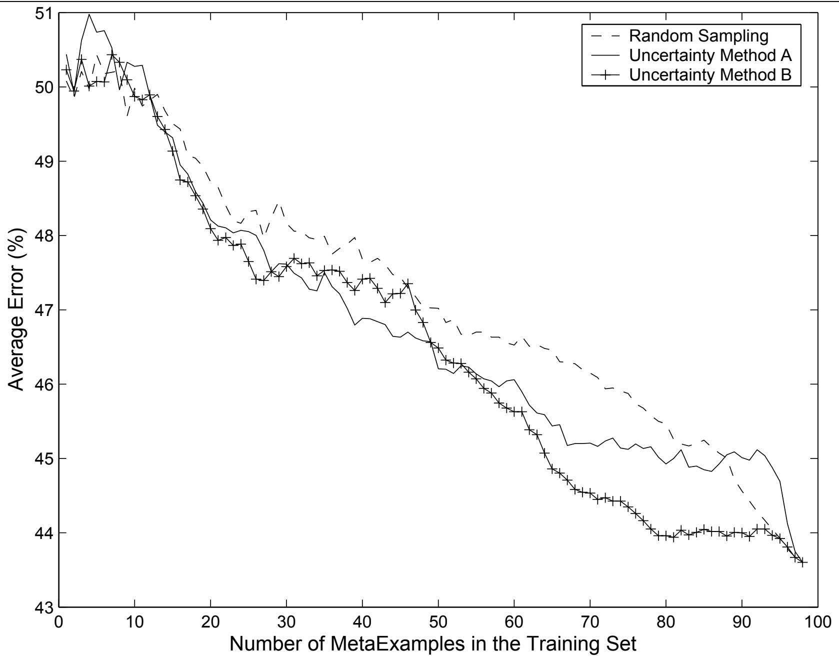

Fig. 2. Case Study I – Average curves of error rates for both Uncertainty methods evaluated in the AL module and for the random sampling method.

statistical gain in 37 points of the curve of error rates, which represent about 38% of the 98 points. These points ranged from 23 to 80 meta-examples, with some observed interruptions (see Fig. 3(a)). The Uncertainty Method B in turn obtained a statistical gain in 44 points of the curve of error rates (about 45% of the 98 points). The statistical gain in this method was more regular compared to the Uncertainty Method A and it was observed especially in the second half of the curve (see Fig. 3(b)). From 54 to 91 included meta-examples, we observed a statistical gain with Uncertainty Method B without interruptions.

5.2. *Case Study II*

In the second case study, the prototype was evaluated in a meta-learning task proposed in [28] which consisted in predicting the performance pattern of Multi-Layer Perceptron (MLP) networks for regression problems. Below, we provide a brief description of the meta-examples related to this task. More details can be found in [28].

The set of meta-examples was generated from the application of MLP to 50 different regression problems, available in the WEKA project.[2] Each meta-example was related to a regression problem and stored: (1) the values of $p = 10$ meta-attributes describing the problem; and (2) a class attribute which indicated the performance pattern obtained by the MLP network on the problem. The set of meta-attributes was composed by:

1. Log of the number of training examples (X_1);
2. Log of the ratio between number of training examples and number of attributes (X_2);
3. Min, max, mean and standard deviation of the absolute values of correlation between predictor attributes and the target attribute (X_3, X_4, X_5 and X_6);
4. Min, max, mean and standard deviation of the absolute values of correlation between pairs of predictor attributes (X_7, X_8, X_9 and X_{10}).

In [28], the meta-examples were assigned to class labels which indicated specific patterns of performance obtained by the MLP. The possible class labels for the meta-examples were defined from a clustering procedure applied on the MLP performance over the 50 prob-

[2]These datasets are specifically the sets provided in the files *numeric* and *regression* available to download in http://www.cs.waikato.ac.nz/ml/weka/.

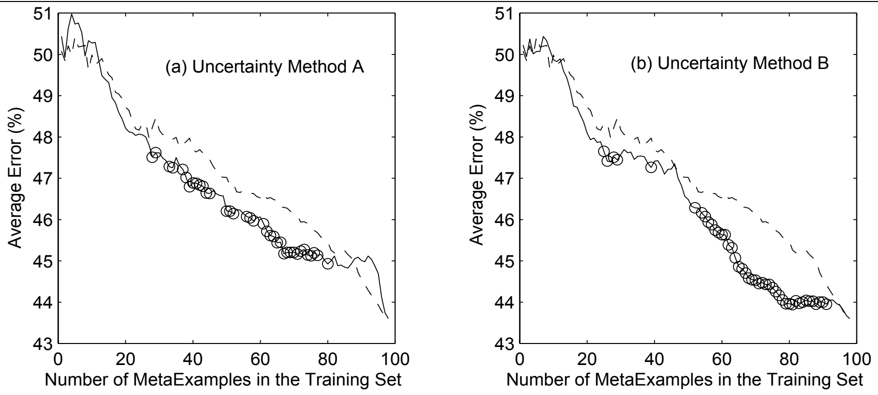

Fig. 3. Case Study I – Comparison of the uncertainty methods to the random sampling in statistical terms. The circles indicate the points in which a statistical gain in performance was observed by using the uncertainty method.

lems. The motivation was to discover groups (clusters) of problems with similar pattern of algorithm performance, and to use the identified clusters as class labels for meta-examples. Such meta-learning approach was adopted, for instance, in [13] (see Section 2). The meta-attributes in this case are used by a meta-learner to predict the specific pattern of algorithms performance for new tasks being presented. The methodology used in [28] to labeling of meta-examples is described below.

Initially, the MLP was empirically evaluated on the 50 regression problems by using two training algorithms, the Backpropagation (BP) [33] and the Levenberg-Marquardt (LM) [17]. Following, the test error rates obtained by using BP and LM on the 50 problems were analyzed by a clustering technique. In this clustering process, the objects to be grouped were points in a bidimensional space, in such a way that each point represented the test error rates obtained by BP and LM on a specific problem. Two clusters of problems were identified and used to define class labels: (1) the *cluster1*, composed by 26 problems, in which both BP and LM algorithms obtained low test errors; and (2) the *cluster2*, composed by 24 problems, in which the LM algorithm obtained intermediate error rates and BP obtained high error rates when used to train the MLP.

5.2.1. Experiments

The experiments performed on this case study followed the same methodology applied in the first case study. We performed 30 runs of experiments for each different value of the parameter k (1, 3, 5, 7, 9 and 11) adopted in the ML module. As in the first case study, the ML module was evaluated by progressively including meta-examples in its training set. The methodology of experiments was applied for both Uncertainty Sampling methods and for the Random procedure. The average curves of error rates were computed across the different values of k and runs of experiments.

5.2.2. Results

As in the first case study, the error rates decreased as the number of meta-examples in the training set increased, considering the three evaluated methods. However, the curves of error rates obtained by both Uncertainty methods were more stable, showing a lower degree of oscillation in the error rates (see Fig. 4), compared to the pattern of results obtained in the first case study. In absolute terms, the results obtained by the Uncertainty methods were also better than the Random method in the most part of the curve of error rates.

The Uncertainty Method A, proposed in [20], obtained lower error rates earlier, when compared to the Uncertainty Method B which is based on entropy. In fact, in the first half of the curve of errors rates the performance of the Uncertainty Method A was steadily better than the entropy-based method. However, from 30 to 35 meta-examples in the training set, there is a turning point in the curves of error rates, in such a way that the Uncertainty Method B became better than method A. The lower performance of Uncertainty Method A in the second half of the curve of error rates was also observed in the first case study. Based on these results, we intend to investigate in future work, the combination of the two methods in order to obtain a consistent performance in the whole curve of error rates.

The good results of the Uncertainty methods were also observed to be statistically significant compared to the Random Sampling (see Fig. 5). A t-test (95% of confidence) applied to the difference of error rates

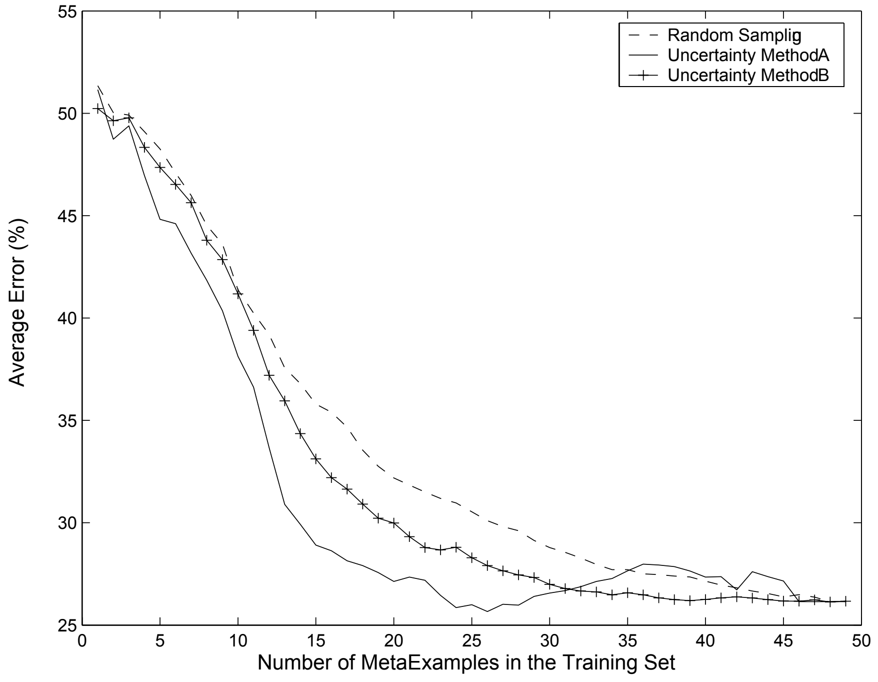

Fig. 4. Case Study II – Average curves of error rates for both the classification uncertainty and the random method.

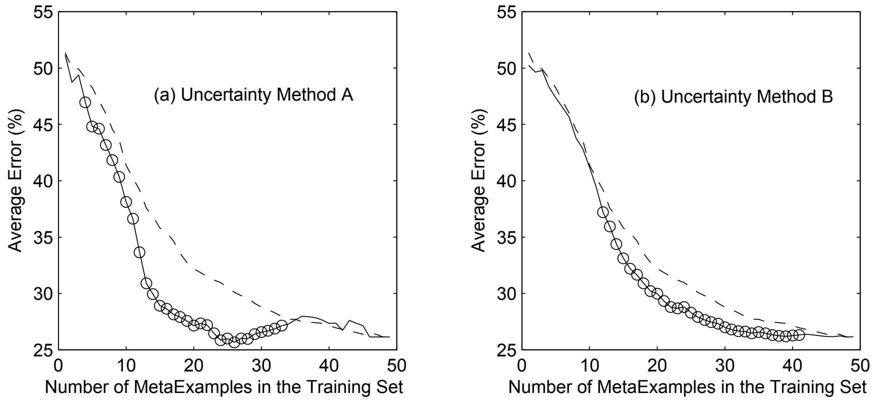

Fig. 5. Case Study II – Comparison of the uncertainty methods to the random sampling in statistical terms. The circles indicate the points in which a statistical gain in performance was observed by using the uncertainty method.

indicated that the Uncertainty Method A obtaining a gain in performance compared to the Random Method in 31 points in the curve of error rates (about 63% of the 49 points). The Uncertainty Method B in turn obtained a statistical gain in performance in 30 points in the curve of error rates (about 61% of the points).

6. Conclusion

In this paper, we presented the proposal of Active Meta-Learning in which Active Learning was used to support the generation on training examples for Meta-Learning. We can point out contributions of our work to two different fields: (1) in the Meta-Learning field, we proposed a new approach to improve meta-learning performance, speeding up the meta-example generation; and (2) in the Active Learning field, we applied its concepts and methods in a context which had not yet been tackled. We envisioned that different developments of the current work will be proposed in the future.

A prototype was implemented using the k-NN algorithm as meta-learner and Uncertainty-based methods for Active Learning. The prototype was evaluated in two different case studies, and the results obtained by the Active Learning methods were significantly better than a Random method for selecting meta-examples.

In future work, we intend to evaluate the use of other Active Learning methods (e.g. committee-based and error-reduction methods), as well as to investigate the combination of different Active Learning methods in the context of Meta-Learning. We also intend to adapt Active Learning methods for other meta-learners which were not deployed in the current work (e.g. Meta-Regression techniques).

Acknowledgments

The authors would like to thank CNPq (Brazilian Agency) for its financial support.

References

[1] N. Abe and H. Mamitsuka, Query learning strategies using boosting and bagging, in: *Proceedings of 15th International Conference on Machine Learning*, 1998, 1–10.

[2] D. Aha, Generalizing from case studies: a case study, in: *Proceedings of the 9th International Workshop on Machine Learning*, 1992, 1–10.

[3] D. Angluin, Queries and concept learning, *Machine Learning* **2** (1998), 319–342.

[4] H. Bensusan and K. Alexandros, Estimating the predictive accuracy of a classifier, in: *Proceedings of the 12th European Conference on Machine Learning*, 2001, 25–36.

[5] P. Brazdil, C. Soares and J. da Costa, Ranking learning algorithms: using IBL and meta-learning on accuracy and time results, *Machine Learning* **50** (2003), 251–277.

[6] R.G. Brown, *Smoothing, Forecasting and Prediction*, Prentice-Hall, Englewood Cliffs, 1963.

[7] D. Cohn, L. Atlas and R. Ladner, Improving generalization with active learning, *Machine Learning* **15** (1994), 201–221.

[8] R. Engels and C. Theusinger, Using a data metric for preprocessing advice for data mining applications, in: *Proceedings of the 13th European Conference on Artificial Intelligence*, 1998, 430–434.

[9] C. Giraud-Carrier, R. Vilalta and P. Brazdil, Introduction to the special issue on meta-learning, *Machine Learning* **54** (2004), 187–193.

[10] A. Kalousis and T. Theoharis, NOEMON: Design, implementation and performance results of an intelligent assistant for classifier selection, *Intelligent Data Analysis* **3** (1999), 319–337.

[11] A. Kalousis and M. Hilario, Feature selection for meta-learning, *Lecture Notes in Computer Science* **2035** (2001), 222–233.

[12] A. Kalousis and M. Hilario, Representational issues in meta-learning, in: *Proceedings of the 20th International Conference on Machine Learning*, 2003, 313–320.

[13] A. Kalousis, J. Gama and M. Hilario, On data and algorithms: understanding inductive performance, *Machine Learning* **54** (2004), 275–312.

[14] C. Koepf, C. Taylor and J. Keller, Meta-analysis: data characterisation for classification and regression on a meta-level, in: *Proceedings of the International Symposium on Data Mining and Statistics*, 2000.

[15] K. Lang and G. Hinton, Time-delay neural network architecture for speech recognition, *CMU Technical Report CS-88-152*, Carnegie-Mellon University, Pittsburgh, 1988.

[16] R. Leite and P. Brazdil, Predicting relative performance of classifiers from samples, in: *Proceedings of the 22nd International Conference on Machine Learning*, 2005, 497–503.

[17] K. Levenberg, A method for the solution of certain non-linear problems in least squares, *Quarterly Journal of Applied Mathematics* **II** (1944), 164–168.

[18] D. Lewis and W. Gale, A sequential algorithm for training text classifiers, in: *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval*, 1994, 3–12.

[19] R. Liere and P. Tadepalli, Active learning with committees for text categorization, in: *Proceedings of the 14th National Conference on Artificial Intelligence* (*AAAI-97*), 1997, 591–596.

[20] M. Lindenbaum, S. Markovitch and D. Rusakov, Selective sampling for nearest neighbor classifiers, *Machine Learning* **54** (2004), 125–152.

[21] P. Melville and R. Mooney, Diverse ensembles for active learning, in: *Proceedings of the 21st International Conference on Machine Learning*,, 2004, 74.

[22] D. Michie, D. Spiegelhalter and D. Taylor, *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, New York, 1994.

[23] I. Muslea, S. Minton and C. Knobrock, Active learning with multiple views, *Journal of Artificial Intelligence Research* **27** (2006), 203–233.

[24] B. Pfahringer, H. Bensusan and C. Giraud-Carrier, Meta-learning by landmarking various learning algorithms, in: *Proceedings of the 17th International Conference on Machine Learning* (*ICML-2000*), 2000, 743–750.

[25] R.B.C. Prudêncio and T.B. Ludermir, Selection of models for time series prediction via meta-learning, in: *Proceedings of the 2nd International Conference on Hybrid Intelligent Systems*, 2002, 74–83.

[26] R.B.C. Prudêncio, T.B. Ludermir and F.A.T. de Carvalho, A modal symbolic classifier to select time series models, *Pattern Recognition Letters* **25** (2004), 911–921.

[27] R.B.C. Prudêncio and T.B. Ludermir, Meta-learning approaches for selecting time series models, *Neurocomputing Journal* **61** (2004), 121–137.

[28] R.B.C. Prudêncio and T.B. Ludermir, Active learning to support the generation of meta-examples, *Lecture Notes in Computer Science* **4668** (2007), 817–826.

[29] R.B.C. Prudêncio and T.B. Ludermir, Active selection of training examples for meta-learning, in: *Proceedings of the 7th International Conference on Hybrid Intelligent Systems*, 2007, 126–131.

[30] H. Raghavan, O. Madani and R. Jones, Active learning with feedback on both features and instances, *Journal of Machine Learning Research* **7** (2006), 1655–1686.

[31] G. Riccardi and D. Hakkani-Tur, Active learning: theory and applications to automatic speech recognition, *IEEE Transactions on Speech and Audio Processing* **13** (2005), 504–511.

[32] N. Roy and A. McCallum, Toward optimal active learning through sampling estimation of error reduction, in: *Proceedings of the 18th International Conference on Machine Learning*, 2001, 441–448.

[33] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning representations by backpropagating errors, *Nature* **323** (1986), 533–536.

[34] I.A. Sampaio, G.L. Ramalho, V. Corruble and R.B.C. Prudêncio, Acquiring the preferences of new users in recommender systems: the role of item controversy, in: *Proceedings of the ECAI 2006 Workshop on Recommender Systems*, 2006, 107–110.

[35] G. Schohn and D. Cohn, Less is more: active learning with support vector machines, in: *Proceedings of the 17th International Conference on Machine Learning*, 2000, 839–846.

[36] H. Seung, M. Opper and H. Sompolinsky, Query by committee, in: *Proceedings of the 6th Annual ACM Conference on Computational Learning Theory*, 1992, 287–294.

[37] C. Shah, Model selection in univariate time series forecasting using discriminant analysis, *International Journal of Forecasting* **13** (1997), 489–500.

[38] C. Soares and P. Brazdil, Zoomed-Ranking: selection of classification algorithms based on relevant performance information, *Lecture Notes in Computer Science* **1910** (2000), 126–135.

[39] L. Todorovski and S. Dzeroski, Combining Classifiers with Meta Decision Trees, *Machine Learning* **50** (2003), 223–249.

[40] S. Tong and D. Koller, Active learning for parameter estimation in Bayesian networks, *Advances in Neural Information Processing Systems* **13** (2000), 647–653.

[41] S. Tong and D. Koller, Support vector machine active learning with applications to text classification, *Journal of Machine Learning Research* **2** (2001), 45–66.

[42] G. Tsoumakas, D. Vrakas, N. Bassiliades and I. Vlahavas, Lazy adaptive multicriteria planning, *Proceedings of the 16th European Conference on Artificial Intelligence*, 2004, 693–697.

[43] A. Venkatachalan and J. Sohl, An intelligent model selection and forecasting system, *Journal of Forecasting* **18** (1999), 167–180.

[44] R. Vilalta and Y. Drissi, A perspective view and survey of meta-learning, *Journal of Artificial Intelligence Review* **18** (2002), 77–95.