Combining Uncertainty Sampling Methods for Supporting the Generation of Meta-Examples

Ricardo B. C. Prudêncio[a], Teresa B. Ludermir[a]

[a] *Centro de Informática, Universidade Federal de Pernambuco*
Recife, Brazil
{rbcp, tbl}@cin.ufpe.br

Abstract

Meta-Learning aims to automatically acquire knowledge relating features of learning problems to the performance of learning algorithms. Each training example in Meta-Learning (i.e. each meta-example) stores features of a learning problem plus the performance obtained by a set of algorithms when evaluated on the problem. Based on a set of meta-examples, a meta-learner will be used to predict algorithm performance for new problems. The generation of a good set of meta-examples can be a costly process, since for each problem it is necessary to perform an empirical evaluation of the algorithms. In a previous work, we proposed the Active Meta-Learning, in which Active Learning was used to reduce the set of meta-examples by selecting only the most relevant problems for meta-example generation. In the current work, we extend our previous research by combining different Uncertainty Sampling methods for Active Meta-Learning, considering that each individual method will provide useful information to select relevant problems. We also investigated the use of Outlier Detection to remove a priori those problems considered as outliers, aiming to improve the performance of the sampling methods. In our experiments, we observed a gain in Meta-Learning performance when the proposed combining method was compared to the individual active methods being combined and also when outliers were removed from the set of problems available for meta-example generation.

Keywords: Meta-Learning, Algorithm Selection, Active-Learning, Uncertainty Sampling, Outlier Detection.

1. Introduction

The selection of adequate algorithms for a given learning problem is an important task to be accomplished by developers in Machine Learning and Data Mining applications [9]. During the algorithm selection, the developer is usually faced to a variety of algorithms to choose, in such a way that selecting the most adequate one is not always a trivial task. Meta-Learning emerged in this context as a framework aiming to relate features of the learning problems to the performance of the learning algorithms [5], thus supporting the selection of the algorithms which best match to the characteristics of the problems at hand.

The knowledge in Meta-Learning is acquired from a set of *meta-examples* associated to learning problems solved in the past by a set of one of more candidate algorithms. Each meta-example is generated from a given problem and stores: (1) features describing the problem (e.g., number of training examples, correlation between attributes,...); and (2) information about the performance obtained by the algorithms when evaluated on the problem (e.g. estimated cross-validation error). By receiving a set of such meta-examples, 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 can then be used to predict algorithm performance for new problems not seen during the Meta-Learning process and to understand the behavior of the algorithms.

Despite the benefits of Meta-Learning, a limitation can be pointed out regarding the process of generating meta-examples. In fact, in order to produce a meta-example, it is necessary to perform an empirical evaluation (e.g. cross-validation) of the candidate algorithms on a problem. Hence, the cost of generating a good enough set of meta-examples may be high, depending, for instance, on the number and complexity of the candidate algorithms, the methodology of empirical evaluation and the size of the available problems. Also, it is not trivial to know a priori how many meta-examples are necessary in order to provide a good learning set for the Meta-Learning process. One could solve this difficulty by generating a very large set of meta-examples, however it is not desirable to spend too much computational resources to produce meta-examples which can eventually be redundant or even irrelevant. A more feasible solution should produce a minimal set of meta-examples only containing information related to relevant learning problems for the Meta-learning task at hand.

In a previous work [24], we proposed the Active Meta-Learning in which Active Learning techniques [8] were used to support the generation of meta-examples. Active Learning is a paradigm of Machine Learning which aims 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 an expensive process, which is the case of Meta-Learning.

In [24], we presented the first experiments performed to evaluate the viability of Active Meta-Learning. In that work, different active methods based on *Uncertainty Sampling* were used to select meta-examples for an instance-based meta-learner. The experiments performed in [24] showed a significant gain in Meta-Learning performance when the Uncertainty Sampling methods were used. However, in these experiments, the performance of the evaluated active methods varied a lot during the selection of meta-examples, and in some cases presented a complementary behavior. Based on these results, in [25] a combining procedure of different active methods (the *Average Rank*) was proposed and evaluated. In this procedure, each method being combined is initially used to generate a *ranking* for the problems available to generate meta-examples. The ranks assigned by the different methods are then averaged in order to provide a final score of relevance for each problem. Finally, the problems with better average ranks are selected to generate new meta-examples. In [25], the Average Rank method yielded better results compared to the individual methods being combined.

In the current work, we extend our previous research by proposing and evaluating new combining procedures, thus performing a further investigation on the combination of different Uncertainty Sampling methods for Active Meta-Learning. In the *Alternate* procedure, a single and different active method is randomly chosen at each active iteration and adopted to perform the selection of meta-examples. In the *Two-Step Tournament* procedure, an individual method is applied to perform an initial selection (filtering) of meta-examples which will be later evaluated by a second active method. Finally, in the *Average Uncertainty*, the uncertainty scores provided by different active methods are normalized and averaged in order to provide the relevance score for each problem. In the current work, we also investigated the effect of outliers in the performance of the Uncertainty Sampling methods, which is a drawback already known in the literature of Active Learning [31]. In this direction, an Outlier Detection technique was used to improve the performance of Uncertainty Sampling by removing outliers among the

problems available for meta-example generation.

In this paper, the techniques being combined were evaluated on a case study which consisted of predicting the performance of Multi-Layer Perceptron (MLP) networks for regression problems. In the performed experiments, the obtained results revealed a gain in the Meta-Learning performance when the combining procedures were used to generate meta-examples. The experiments also revealed that the performance of Active Meta-Learning can be improved with the removal of problems considered as outliers.

Section 2 brings a brief presentation of Meta-Learning, followed by section 3 which presents relevant work on Active Learning. Section 4 presents the proposal of Active Meta-Learning, followed by the techniques combined in our work. Section 5 presents the experiments and results. Finally, section 6 concludes the paper.

2. Meta-Learning

Meta-Learning tries to understand the performance of learning algorithms based on experience systematically acquired from different problems [5, 11]. It acquires knowledge from a set of meta-examples, which store the experience obtained from applying the algorithms to different problems in the past. According to [39], Meta-Learning can be defined by considering four aspects:

- (1) the problem space, P, representing the set of instances of a given problem class (usually classification and regression problems);
- (2) the meta-feature space, F, that contains characteristics used to describe the problems (e.g., number of training examples, correlations between attributes, ...);
- (3) the algorithm space, A, that is a set of one or more candidate algorithms to solve the problems in P;
- (4) a performance information, Y, that characterizes the performance of an algorithm on a problem (e.g., classification accuracy estimated by cross-validation).

In this framework, Meta-Learning receives as input a set of meta-examples, in which each meta-example is derived from the empirical evaluation of the algorithms in A on a given problem in P. More specifically, each meta-example

stores: (1) the values of the meta-features F extracted from a problem; and (2) the performance information Y estimated for the problem. Hence, the *meta-learner* is another learning technique that relates a set of predictor attributes to the performance information.

Different Meta-Learning approaches have been proposed in the literature. In a strict approach (e.g., [15]), each meta-example stores as performance information a class attribute associated to the candidate algorithm which obtained the highest accuracy in the learning problem. In this formulation, the meta-learner just becomes a classifier in which the meta-features correspond to predictor attributes for the class label associated to the best candidate algorithm.

Other approaches have been proposed in order to add new functionalities in the Meta-Learning process and to provide new variants of the algorithm selection task. The Meta-Regression approach [3], for instance, tries to directly predict the accuracy (or alternatively the error) of the learning algorithms instead of simply predicting the class that corresponds to the best algorithm. In [12, 6, 27], the authors proposed different Meta-Learning approaches to generate rankings of algorithms taking into account multiple performance information criteria. We can also mention the use of Meta-Learning for selecting parameters of a single learning algorithm, including for instance the selection of kernel parameters for Support Vector Machines [41] and the use of Meta-Learning to decide when to prune a decision tree [40]. More detailed reviews of these developments can be found in recent textbooks [5, 13, 11].

Although Meta-Learning has been mainly investigated for algorithm selection on classification and regression problems, its concepts and techniques have also been adapted to other domains of application, including time series forecasting [23], combinatorial optimization [38], software engineering [7] and bioinformatics [21, 42]. In this sense, as highlighted in [39], Meta-Learning can be useful in 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. Active Learning

Active Learning is a paradigm of Machine Learning in which the learning algorithm has some control over the inputs on which it trains [8]. 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 [18], text classification [46, 29], speech recognition [30] and information filtering [32]. Also, according to [4], Active Learning presents advantages compared to the traditional *passive learning*, since it can be more efficient in case of time-consuming learning algorithms and can improve generalization performance by selecting only the informative training examples.

In [2], the author proposed the *membership query* approach for Active Learning, in which, the learner artificially creates informative examples in the input domain and asks the teacher to annotate it. Although it represents a landmark work in the Active Learning field, the membership query 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) [20].

According to [18], 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 [20], the selective sampling methods can be distinguished on three main categories, *uncertainty-sampling* methods, *version space reduction* methods and *error reduction* methods, described just below.

The concept of uncertainty has been adopted in different contexts to improve learning performance [47]. In *uncertainty-sampling* methods [16, 34, 33, 18, 29, 35, 48] 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 [20], these methods can be straightforwardly applied to many different learning algorithms. Among the uncertainty measures proposed in the literature, we can cite distance-based measures for instance-based learning [18], entropy-based measures [35], margin-based measures [33, 37] and maximal classification ambiguity [48]. A limitation of uncertainty based methods, however, is that they often select examples that are outliers [31]. 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, 17, 19, 36, 46], 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 [18, 31, 45], 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 [31].

4. Active Meta-Learning

As it was said, Active Learning methods have been applied to reduce the costs associated to data acquisition in different Machine Learning applications. In our work, we argue that Active Learning can also be useful in the context of Meta-Learning, since the process of producing an adequate set of meta-examples may be costly. In fact, in order to generate a single meta-example from a given problem, it is necessary to perform an empirical evaluation of the candidate algorithms on the problem (as explained in section 2). Hence, the cost of generating a whole set of meta-examples may be high depending on a number of aspects, including for instance, the methodology adopted for empirical evaluation 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. Active Learning can be specially useful in the context of new architectures developed to collect and exploit meta-knowledge through large databases of experiments in Machine Learning [28].

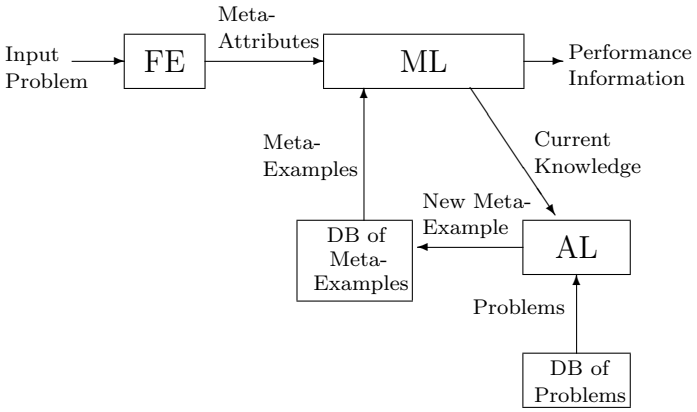

Figure 1: System Architecture.

The use of Active methods for supporting the generation of meta-examples can not only reduce the costs of data acquisition but can also improve the generalization performance of the Meta-Learning process. In the literature of Meta-Learning, training examples are usually generated from learning problems which are randomly collected in data repositories. Hence, the collected problems can eventually be irrelevant or redundant for the Meta-learning task at hand. By deploying Active Learning, we expected to select only the most relevant problems, thus producing a high quality set of meta-examples and improving performance in the Meta-Learning task.

Fig. 1 presents the architecture of system following the Active Meta-Learning originally proposed in [24]. This proposal 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 [24], we presented the initial experiments performed to evaluate the Active Meta-Learning proposal. In this work, two different Uncertainty Sampling methods were evaluated in an implemented prototype. Experiments performed in case studies revealed that no active method was always better than the other. We also observed in our experiments that despite the gain achieved in meta-learning performance, the Uncertainty Sampling often generated meta-examples which are outliers. As mentioned in section 3, such limitation of Uncertainty Sampling was already stated in the literature of Active Learning.

In the current work, we proposed to combine different active methods, aiming to have a better assessment of the relevance of each learning problem available for the generation of meta-examples. More specifically, we combined in the current work different Uncertainty Sampling techniques and an Outlier Detection procedure, applied to Active Meta-Learning. In this combination, we first remove unlabeled meta-examples considered as outliers, and then combine different Uncertainty Sampling techniques in order to progressively select from the remaining unlabeled meta-examples the most informative ones to be labeled. Preliminary results of combining different techniques for Active Meta-Learning were presented in [25], in which a combining method based on *Average Rank* of unlabeled meta-examples was proposed. In the current work, we extend our previous research by proposing and evaluating three new combining methods: the *Alternate*, the *Average Uncertainty* and *Two-Step Tournament* methods (described in Section 4.2). We highlight that to the best of our knowledge the combining methods proposed here are original in the literature. We also provide in the current work a better description and discussion of the proposed techniques as well as more comparative experiments, specifically aiming to evaluate the combined methods compared to the individual methods being combined and also to evaluate the value of the outlier detection technique.

In order to evaluate our proposal, we implemented a prototype which was applied in a case study. In this prototype, the k-Nearest Neighbors (k-

NN) algorithm was used to predict the performance of MLPs for regression problems. A distance-based technique [14] was adopted to remove outliers, and two different Uncertainty Sampling methods were used in isolation and also being combined by the four combining methods considered. In the next subsections, we provide more details of the implemented prototype.

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 been applied in different applications of Meta-Learning [6, 41, 27]. According to [6], instance-based learning algorithms such as the k-NN have some advantages when applied to Meta-Learning. For instance, when a new meta-example becomes available, it can be easily integrated without the need to initiate re-learning [6]. Also, the k-NN algorithm may be interesting for active learning purposes, since it may be able to produce reliable results with few training examples. In this 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. Following previous work in the meta-learning field (e.g. [6]), the distance function (*dist*) implemented in the prototype was the unweighted L_1-Norm, defined as:

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

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

As seen, the Meta-Learner acquires knowledge from a set of labeled meta-examples associated to a set of learning problems. The Active Learning module, described in this section, receives a set of unlabeled meta-examples, associated to the problems in which the candidate algorithms were not yet evaluated and, hence, the class labels are not known. Therefore, the main objective of this module is to incrementally select unlabeled meta-examples to be labeled.

As said, in this module we combined two classes of techniques. First, an Outlier Detection technique is used to remove unlabeled meta-examples which are considered as spurious points in the meta-learning task. Following, an Uncertainty Sampling method is used to select from the remaining set of unlabeled meta-examples, the most informative ones to be labeled. Details of the two techniques are described as follows.

4.2.1. Outlier Detection

In our prototype, we adapted the Distance-Based method proposed in [14] for Outlier Detection. Here, we eliminated from the set of the unlabeled meta-examples, those ones which most deviates from the others in terms of its distances.

Let $\widetilde{E} = \{\widetilde{e}_1, \ldots, \widetilde{e}_m\}$ be the set of m problems associated to the available set of m unlabeled meta-examples $\widetilde{ME} = \{\widetilde{me}_1, \ldots, \widetilde{me}_m\}$. Each $\widetilde{me}_i \in \widetilde{ME}$ stores the description $\widetilde{\mathbf{x}}_i$ of a problem $\widetilde{e}_i \in \widetilde{E}$. For detecting outliers, we first calculate the average distance between the meta-examples in $\widetilde{ME}$. Formally, for each different pair $(\widetilde{me}_i, \widetilde{me}_j)$, we first calculate the distance: $dist(\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{x}}_j)$. Following, we compute the average of these distances as follows:

$$\mu_{dist} = \frac{1}{m * (m-1)} \sum_{\widetilde{me}_i, \widetilde{me}_j \in \widetilde{ME}, i \neq j} dist(\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{x}}_j) \quad (2)$$

Finally, in order to measure how much an unlabeled meta-example $\widetilde{me}_i$ is considered as an outlier, we compute the proportion of the other unlabeled meta-examples in $\widetilde{ME}$ which are distant from it by at least the reference value μ_{dist}. Formally, let $G_i = \{\widetilde{me}_j \in \widetilde{ME} \,|\, i \neq j, dist(\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{x}}_j) \geq \mu_{dist}\}$ be the set of unlabeled meta-examples which are distant from $\widetilde{me}_i$. The measure $OutlierDegree$ is defined as:

$$OutlierDegree(\widetilde{me}_i) = \frac{|G_i|}{m-1} \tag{3}$$

The unlabeled meta-examples can be sorted by using this measure, in such a way that the meta-examples with the highest values of $OutlierDegree$ are considered as outliers. In our prototype, the top 10% of unlabeled meta-examples in this ranking are removed from the set of candidates to generate labeled meta-examples.

4.2.2. Uncertainty Sampling

In our prototype, we evaluated different methods for uncertainty sampling. The first two methods described in this section deployed different criteria for assigning degrees of classification uncertainty to the unlabeled meta-examples. The next four methods deploy different combining strategies, aiming to explore the eventual advantages of the first two methods.

(a) Uncertainty Sampling: Distance-Based Method

The uncertainty of k-NN was defined in [18] 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. 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 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 E, the classification uncertainty of k-NN for each $\widetilde{e} \in \widetilde{E}$ is defined as:

$$\mathcal{S}(\widetilde{e}|E) = \frac{\min_{e_i \in E} dist(\widetilde{\mathbf{x}}, \mathbf{x}_i)}{\sum_{l=1}^{L} \min_{e_i \in E_l} dist(\widetilde{\mathbf{x}}, \mathbf{x}_i)} \quad (4)$$

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:

$$\widetilde{e}^* = argmax_{\widetilde{e} \in \widetilde{E}} \mathcal{S}(\widetilde{e}|E) \quad (5)$$

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 available data of the problem.

We highlight that the above method may be biased towards the number of classes, since the values of $\mathcal{S}(\widetilde{e}|E)$ tend to decrease as more distances are used in Eq. (4). In this way, the performance of this method may be harmed in the presence of many class labels since the uncertainty measure tends to be less discriminative. For more than two classes, we suggest here an alternative definition of $\mathcal{S}(\widetilde{e}|E)$. Let c_l be the class label associated to the labeled meta-example nearest to $\widetilde{e}$. The uncertainty $\mathcal{S}(\widetilde{e}|E)$ can be defined as:

$$\mathcal{S}(\widetilde{e}|E) = \frac{\min_{e_i \in E} dist(\widetilde{\mathbf{x}}, \mathbf{x}_i)}{\min_{e_i \in E} dist(\widetilde{\mathbf{x}}, \mathbf{x}_i) + \min_{e_i \in \{E - E_l\}} dist(\widetilde{\mathbf{x}}, \mathbf{x}_i)} \quad (6)$$

The above equation is similar to Eq. (4), however in the denominator we only consider the sum of the distance to the nearest labeled neighbor and the distance to the nearest labeled example from any other class (instead of using all classes). Both formulations lead to the same results for two classes, but Eq. (6) is not biased in case of a higher number of classes.

(b) Uncertainty Sampling: Entropy-Based Method

In the second active method, 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:

$$p_C(\widetilde{e}|E) = (p(C(\widetilde{e}) = c_1|E), \ldots, p(C(\widetilde{e}) = c_L|E)) \quad (7)$$

According to [44], the entropy of the probability distribution reflects the uncertainty of the classifier in the predicted class value. The entropy of the probability distribution is computed as:

$$Ent(\widetilde{e}|E) = -\sum_{l=1}^{L} p(C(\widetilde{e}) = c_l|E) * \log_2 p(C(\widetilde{e}) = c_l|E) \quad (8)$$

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:

$$\widetilde{e}^* = argmax_{\widetilde{e} \in \widetilde{E}} Ent(\widetilde{e}|E) \quad (9)$$

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.

(c) Combining Method: Average Rank

In this section, we described a method to combine the uncertainty criteria described in the previous subsections. The combining method was adapted from a previous work [25] which used the concept of *average ranks* originally applied to active construction of user profiles in recommending systems [43].

In the combination, each uncertainty method being combined is initially used to generate a ranking of unlabeled meta-examples. Following, the rankings provided by the different methods are averaged and the unlabeled meta-example with the best average rank is selected. The combining method can be formally described as follows.

Let $r^A(\widetilde{e})$ and $r^B(\widetilde{e})$ be the ranks of the unlabeled meta-example $\widetilde{e}$, by respectively considering the uncertainty methods A and B (i.e., by using the measures $\mathcal{S}(\widetilde{e}|E)$ and $Ent(\widetilde{e}|E)$ to sort the meta-examples in a decreasing order and to assign the respective ranks). Hence, the lower is the rank of a meta-example, the higher is its uncertainty. The two ranks can be combined using the equation:

$$\overline{r}(\widetilde{e}) = w^A * r^A(\widetilde{e}) + w^B * r^B(\widetilde{e}) \quad (10)$$

where $w^A, w^B \in [0;1]$ and $w^A + w^B = 1$. In this equation, $\overline{r}(\widetilde{e})$ is the average rank of $\widetilde{e}$, weighted by the numerical values w^A and w^B. Finally, the unlabeled meta-example $\widetilde{e}^* \in \widetilde{E}$ with lowest average rank is selected:

$$\widetilde{e}^* = argmin_{\widetilde{e} \in \widetilde{E}} \overline{r}(\widetilde{e}|E) \tag{11}$$

We highlight that the combining method described above can be easily generalized to combine more than two methods. Also, different values can be assigned to w^A and w^B in order to control the importance of each method being combined. In our work, for simplicity, we define $w^A = w^B = 0.5$, although different configurations of weights can be evaluated in the future.

(d) Combining Method: Alternate

In the previous combining method, at each iteration an available unlabeled meta-example is selected by processing information provided by all active methods being combined. A simpler combining method is presented in this section where at each iteration a single method is randomly chosen from the pool of active methods and then it is adopted to select meta-examples. We named this combining procedure as *Alternate* method. In our case, we have two active methods as candidates and only one method is randomly selected with equal probability (50%) at each active learning iteration.

The Alternate method may be less adequate than the previous combining method. The information provided by the active methods is not *actually* combined by the Alternate method in order to sort the most relevant meta-examples. Despite this limitation, the Alternate method is computationally less expensive than the Average Rank since it is not necessary to run all active learning methods at each iteration (only one active learning method is used at a time).

(e) Combining Method: Two-Step Tournament

Given two active learning methods, the combining method described in this section performs a two-step tournament to choose the unlabeled meta-examples. Initially, a chosen active learning method is applied to perform a preliminary selection of q unlabeled meta-examples. These unlabeled meta-examples are sorted by the second active learning method and the best ranked meta-example is then selected.

Formally, the first selection step performed by a method A results on a set $\widetilde{E}_A$ of q unlabeled meta-examples. By using a second method B, an

unlabeled meta-example is selected as follows:

$$\widetilde{e}^* = argmin_{\widetilde{e} \in \widetilde{E}_A} r^B(\widetilde{e}|E) \tag{12}$$

In the above equation, $r^B(\widetilde{e}|E)$ is the rank of the unlabeled meta-example $\widetilde{e}$ by considering the uncertainty method B. In our work, at each iteration we randomly adopt a different active learning method at each Tournament step (with equal probabilities). The Two-Step Tournament can be adapted to combine more than two methods by randomly selecting two methods at a time to perform the Tournament steps.

(f) Combining Method: Average Uncertainty

This combining method is similar to the Average Rank method, however, the meta-examples are evaluated by averaging the uncertainty values provided by the methods being combined. Initially, the uncertainty values provided by the active learning methods for the unlabeled meta-examples are computed and then normalized. Formally, let $P^A(\widetilde{e})$ and $P^B(\widetilde{e})$ be the normalized uncertainty value provided respectively by methods A and B for the unlabeled meta-example $\widetilde{e}$. By adopting the measures $\mathcal{S}(\widetilde{e}|E)$ and $Ent(\widetilde{e}|E)$ the normalized uncertainty values are:

$$P^A(\widetilde{e|E}) = \frac{\mathcal{S}(\widetilde{e}|E)}{\sum_{\widetilde{e}_i \in \widetilde{E}} \mathcal{S}(\widetilde{e}_i|E)} \tag{13}$$

$$P^B(\widetilde{e|E}) = \frac{Ent(\widetilde{e}|E)}{\sum_{\widetilde{e}_i \in \widetilde{E}} Ent(\widetilde{e}_i|E)} \tag{14}$$

The average normalized uncertainty is computed as:

$$\overline{P}(\widetilde{e}) = w^A * P^A(\widetilde{e}) + w^B * P^B(\widetilde{e}) \tag{15}$$

where $w^A, w^B \in [0; 1]$ and $w^A + w^B = 1$ (in our work, we adopted $w^A = w^B = 0.5$). The unlabeled meta-example $\widetilde{e}^* \in \widetilde{E}$ with lowest average normalized uncertainty is selected:

$$\widetilde{e}^* = argmin_{\widetilde{e} \in \widetilde{E}} \overline{P}(\widetilde{e}|E) \tag{16}$$

5. Case Study

In the case study, the prototype was evaluated in a meta-learning task which consisted in predicting the performance of Multi-Layer Perceptron (MLP) networks for regression problems. The MLP network is a widespread learning model adopted in different applications over the years. Predicting the performance of MLP can be useful for practitioners and developers of learning applications.

The dataset of meta-examples adopted in the current work was originally built in [24] and later used in [25] for a preliminary evaluation of active learning combining methods. In the current work, we decide to maintain this dataset in order to perform a consistent comparison to our previous research. Also, the regression problems used to produce the meta-examples present a large variability in its features. This variability has shown to be convenient to Meta-Learning studies, since different patterns of learning performance are observed depending on the problem being solved. In this section below, we provide a description of the meta-examples adopted in our work.

The set of meta-examples was generated from the application of MLP to 50 different regression problems, available in the WEKA project[1]. 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 categorized the performance obtained by the MLP network on the problem.

The first step to generate a meta-example from a problem is to extract its meta-features. In the case study, a total number of $p = 10$ meta-features adopted in [24] was used to describe the datasets of regression problems:

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

[1]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/

The meta-feature X_1 is an indicator of the amount of data available for training, and X_2, in turn, indicates the dimensionality of the dataset. The meta-features X_3, X_4, X_5 and X_6 indicate the amount of relevant information available to predict the target attribute. The meta-features X_7, X_8, X_9 and X_{10}, in turn, indicate the amount of redundant information in the dataset.

The second step to generate a meta-example is to estimate the performance information on the problem being tackled. In our case study, this step consists of evaluating the performance of one-hidden layer MLPs trained by the standard BackPropagation (BP) algorithm[2]. We highlight here that there are different algorithms to train MLPs that could have been used to improve MLP's performance. However, the aim in this work is not to achieve the best possible performance with MLPs but to predict learning performance. Other learning algorithms to train MLPs (such as Extreme Learning Machines [10]) can be applied in the future as new case studies.

From the MLP evaluation, we produce the performance information stored in the meta-examples which will correspond to a label C indicating the degree of success when the MLP was applied to the problem. In order to define the labels of each problem, the following methodology of evaluation was applied.

The problem's dataset was divided in the training, validation and test sets, in the proportion of 50%, 25% and 25%. As usual, the training set was used to adjust the MLP's weights, the validation set was used to estimate the MLP performance during training, and the test set was used to evaluate the performance of the trained MLP. The optimal number of hidden nodes was defined by testing the values 1, 2, 4, 8, 16 and 32. For each number of nodes, the MLP was trained 10 times with random initial weights. In the training process, we adopted benchmarking rules [22]: early stopping was used to avoid overfitting with the GL_5 stopping criterion and a maximum number of 1000 training epochs (see [22] for details of these rules). The optimal number of nodes was chosen as the value in which the MLP obtained the lowest average NMSE (Normalized Mean Squared Error) on the validation set over the 10 runs. The NMSE is defined as:

$$NMSE = \frac{\sum_{i=1}^{n_v}(t_i - o_i)^2}{\sum_{i=1}^{n_v}(t_i - \bar{t})^2} \tag{17}$$

[2]The BP algorithm was implemented by using the NNET Matlab toolbox. Learning rates were defined by default.

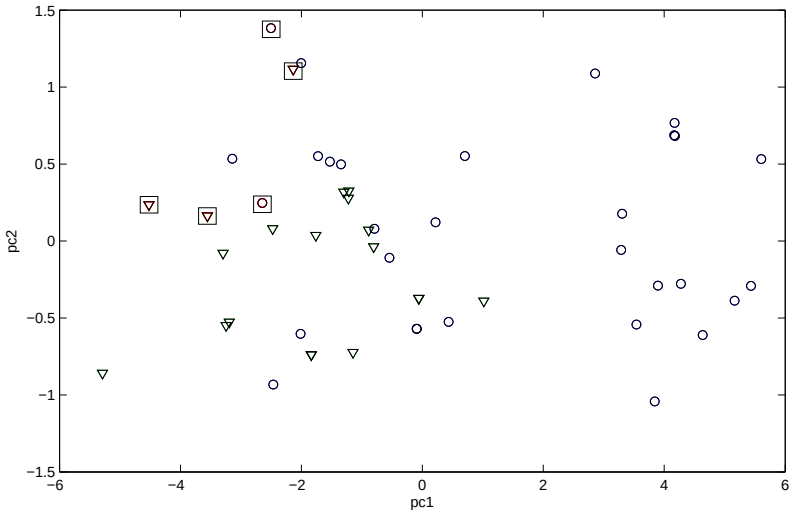

Figure 2: Visualization of the meta-examples using a PCA projection. Circles correspond to meta-examples from class label 0 and triangles correspond to meta-examples from class label 1. Possible outliers are indicated by the squared points.

In the equation, n_v is the number of examples in the validation set, t_i and o_i are respectively the true and the predicted value of the target attribute for example i, and $\bar{t}$ is the average of the target attribute. The NMSE values have no scale and are comparable across different datasets, which is adequate to Meta-Learning [41]. Values of NMSE lower than 1 indicate that the MLP provided better predictions than the mean value at least. Finally, the performance information C related to a problem was defined in our prototype as a binary attribute assigned to 1 if the observed NMSE is lower than 0.5 and assigned to 0 otherwise. Hence, the meta-examples with class label 1 were those problems in which the MLP obtained the best performance patterns.

Fig. 2 presents a 2-dimensional visualization of the set of meta-examples by using a PCA projection. This figure indicates that the set of meta-examples actually presents some class structure that may be identified by a learning algorithm. Despite of the observed structure, the PCA projection also suggests challenging features, such as non-separability of the classes and no clear class boundaries, which makes this meta-learning task a non-

trivial learning problem. Fig. 2 also presents the possible outliers indicated by the method presented in Section 4.2.1. This projection does not suggest that these examples are strong outliers. However, it could suggest that the identified points are more periferical than average, and hence they could be eliminated without compromising learning performance. In fact, as it will seen in the experiments, the elimination of such points in fact improved the performance of the active learning methods.

In the next subsections, we present the experiments performed in our work motivated by different aims. First, we intended to evaluate whether the active methods considered in our work were useful to overcome a passive (random) procedure for selecting meta-examples. Also, we evaluated the performance of the proposed combining methods compared to the individual methods being combined (section 5.1- Experiments I). Second, we intended to verify the utility of the outlier detection mechanism in improving the performance of active learning process for the active methods considered (section 5.2- Experiments II).

5.1. Experiments I

The prototype was evaluated for different configurations of the k-NN meta-learner (with k = 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 49 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 49 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 49 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 50 steps of the leave-one-out experiment.

The above procedure was applied for each uncertainty sampling method considered in the AL module (see section 4.2.2). 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

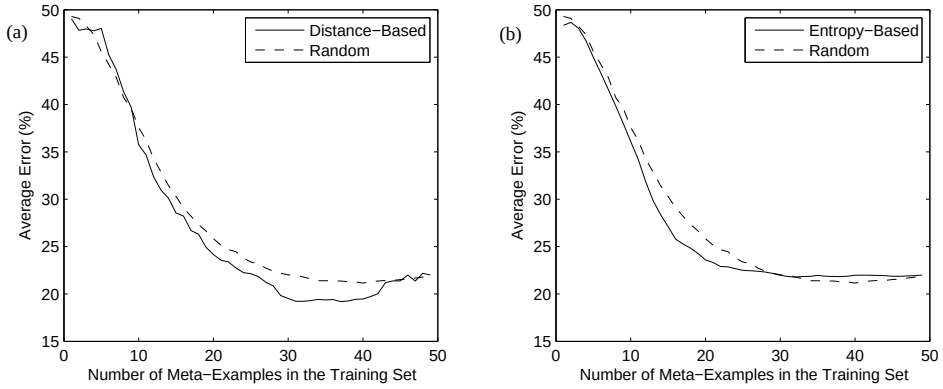

Figure 3: Curve of error rates: (a) Distance-Based method vs. Random method; (b) Entropy-Based method method A vs. Random method

problems. According to [18], 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 100 different runs for each configuration of the k-NN meta-learner.

Fig. 3 shows the curves of error rate of each uncertainty method compared to the curve obtained by using Random Sampling. In the most part of the curves, the error rates achieved by using the uncertainty methods were lower than the rates observed by using the random procedure. The Distance method and the Entropy method achieved lower error rates compared to the Random Sampling in 40 and 32 points in the curves respectively (about 81.6% and 65.3% of the 49 points). The good results of the uncertainty methods were also observed to be statistically significant especially for Distance method. In fact, a t-test (95% of confidence) applied to the difference of error rates indicated that the Distance method obtaining a gain in performance compared to the Random Method in 28 points in the curve of error rates (about 57.14% of the 49 points). The Entropy method in turn obtained a statistical gain in performance in 13 points in the curve of error rates (about 26.5% of the points).

The Distance method and the Entropy method yielded different performance patterns considering different segments of the error curves. In fact, in the first half of the curves the Entropy method achieved a better comparative performance. However, there is a turning point in the second half of

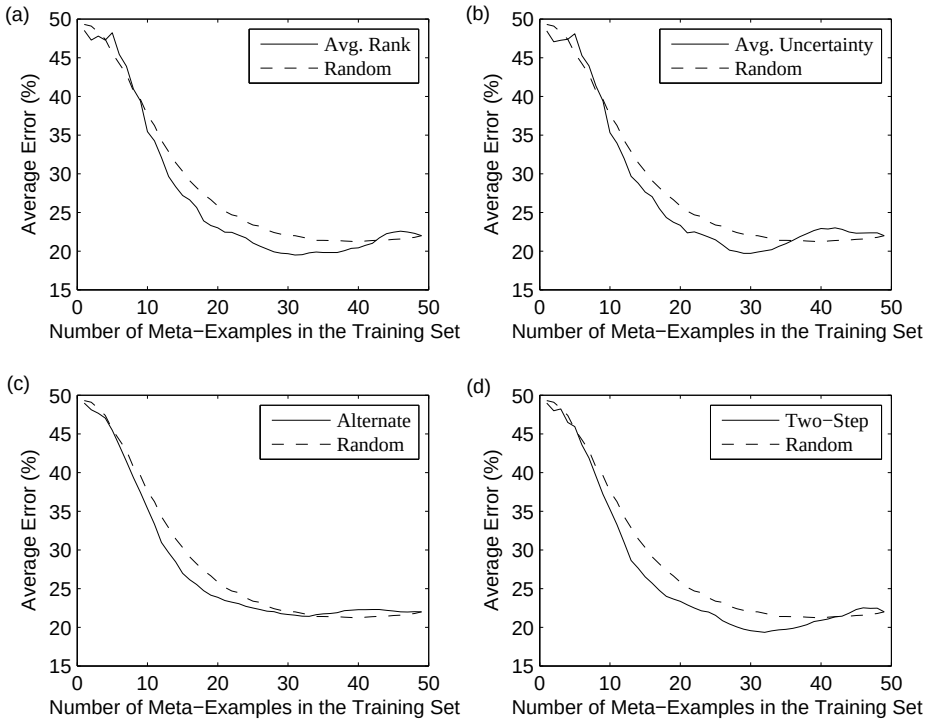

Figure 4: Curve of error rates: Combining methods vs. Random method

Table 1: Average ranks obtained by all active methods along the curves of error rates

	Average rank	Standard deviation
Two-Step	2.5714	1.2910
Avg. Rank	2.9796	1.6136
Distance	3.4694	2.1223
Avg. Uncert.	3.7959	1.6581
Alternate	3.9184	1.4838
Entropy	3.9388	1.7249

the curves of error rates, in such a way that the Distance method became better than Entropy method. This result motivates the use of combining procedures that would take advantage of different algorithms. The viability of using combining procedures was in fact confirmed in our experiments (see Fig. 4). The Average Rank, Average Uncertainty, Alternate and Two-Step methods were better than the Random method respectively in 38, 30, 33 and 41 points of the error curves (about 77.5, 61.2, 67.3% and 83.6% of the curves). Concerning statistical gain, the Average Rank, Average Uncertainty, Alternate and Two-Step methods were better than the Random method in 30, 25, 15 and 30 respectively (corresponding to 61.2%, 53.1%, 30.6% and 61.2% of the curves).

The combination of active methods yielded a more consistent performance along the curve especially when the Two-Step and the Average Ranking were adopted. Table 1 presents the average ranks obtained by all active methods at each point of the curves (i.e., rank = 1 for the best method, rank = 2 for the second method, and so on). The Two-Step obtained the best average rank over the 49 points, followed by the Average Rank method. Both procedures were better than the individual components being combined (i.e. better than both Distance and Entropy). We also observe here that the standard deviations obtained by Two-Step and Average Rank were lower than the standard deviations observed for Distance and Entropy. The remaining combining procedures (Average Uncertainty and Alternate) were better than Entropy but worse than the Distance method. These results indicate that in general the combining methods obtained better positions compared to its individual components.

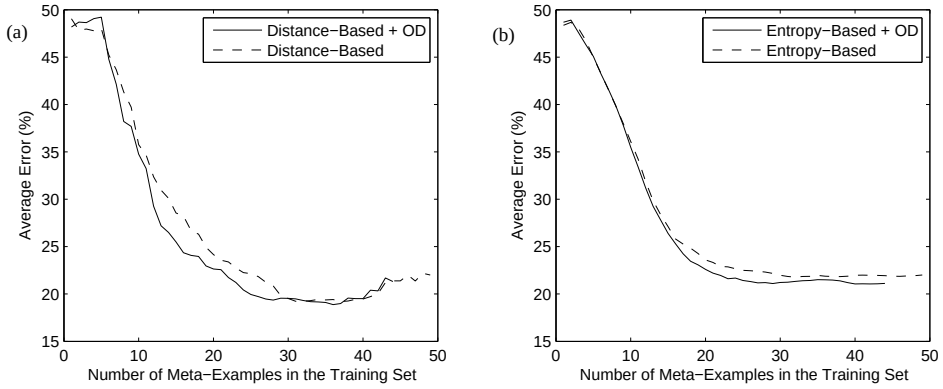

Figure 5: Curve of error rates: Distance and Entropy methods with and without Outlier Detection

Table 2: Average ranks obtained by all active methods along the curves of error rates - with the use of outlier detection

	Average rank	Standard deviation
Two-Step	2.2273	1.2915
Avg. Rank	2.8182	1.4350
Distance	3.3864	1.7943
Avg. Uncert.	3.5909	1.5450
Alternate	4.3409	1.2378
Entropy	4.5909	1.7430

5.2. Experiments II

In this step of experiments, we evaluated the usefulness of Outlier Detection to improve the Uncertainty Sampling methods. We followed the same methodology of experiments as described in the previous experiments, however at each step of leave-one-out, 5 candidate meta-examples (about 10% of the meta-examples) were initially removed by using the Outlier Detection technique. Following, the Uncertainty Sampling method incrementally included one meta-example in the training set of the Meta-Learner, up to the total number of 44 training meta-examples.

Fig. 5 shows the curves of error rate of each uncertainty method (Dis-

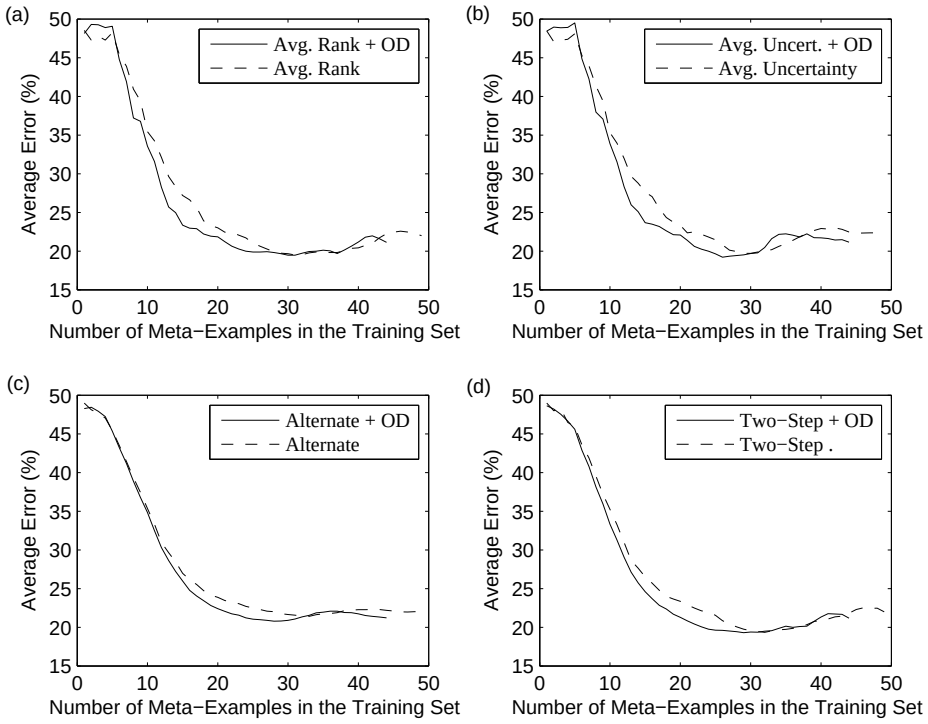

Figure 6: Curve of error rates: Combining methods with and without Outlier Detection

tance and Entropy methods), with and without the removal of outliers. The Distance and the Entropy methods achieved lower error rates in 31 and 39 points in the curves respectively (about 70.45% and 88.64% of the 44 points) when the Outlier Detection technique was applied. By applying a t-test (95% of confidence), the improvement in performance obtained with the use of Outlier Detection was statistically verified for the Distance method in 18 points of the curve of error rates. For the Entropy method, the improvement in performance has shown to be significant in 15 points.

Similar results were obtained when the combining methods were applied with and without outlier detection (see Fig. 6). The Average Rank, Average Uncertainty, Alternate and Two-Step methods obtained better performance respectively in 30, 35, 36 and 30 points when outlier detection was adopted (or 68.18%, 81.82%, 79.55% and 68.18% of the curves). For the combining methods, the improvement in performance was statistically verified in 18, 24, 19 and 18 points, which is similar to the good results observed for the individual methods being combined. We also observed that in general, when the outliers were removed, each active method achieved its respective best error rate in earlier points of the curves. The results observed in these experiments indicate that the use of Outlier Techniques can improve the performance of Active Meta-Learning.

Finally, the combining methods yielded a more consistent performance along the curve compared to the methods being combined. The best ranks of methods along the curves of error rates were achieved by the Two-Step method, followed by the Average Rank method (see Table 2). The other combining methods were worse than the Distance method, but they were better than the Entropy method. These results are similar to the results obtained in the first experiments without the removal of outliers.

6. Conclusion

In this paper, we presented the proposal of combining different Uncertainty Sampling techniques for Active Meta-Learning. In this work, different combining procedures were proposed and evaluated. Experiments were performed in a case study by combining two different Uncertainty Sampling methods. The combining methods obtained in general better results compared to the individual methods being combined. Combining methods obtained the best average positions along the curve of error rates, indicat-

ing that they were more robust regarding the number of generated meta-examples.

In our work, we also evaluated the use of Outlier Detection to improve the performance of the Uncertainty Sampling Methods. The experiments revealed that the active methods were in fact sensitive to the presence of outliers, in such a way that its performance was improved when the Outlier Detection technique was applied.

In future work, we intend to investigate the combination of a higher number of active methods, including the version space and error reduction methods not considered yet. Some parameters of the combining methods can be changed in the future. For instance, in the current work, we combined the active methods by Average Rank and Average Uncertainty using an equal weight to each method, thus assuming that each method is equally important in the combination. In future work, we intend to propose procedures to adapt the combining weights in order to assign different contributions for each method.

We highlight that all conclusions made in the current work are related to meta-learning defined as a classification task. However, there are other meta-learning definitions that could be deployed (as seen in Section 2). For instance, in the Meta-Regression approach, a meta-learner tries to directly predict the numerical performance of the candidate algorithms, instead of predicting class labels associated to the meta-examples. Although less common in the meta-learning literature, this approach can produce interesting outcomes to perform algorithm selection. Previous research has been already devoted to active learning for Meta-Regression [26], however a deeper investigation has to be done in order to verify whether the combination of different active methods would work in this task.

Finally, we highlight that in spite of the statistical significance of results observed in our case study, the performance achieved by the developed methods are still on the border of statistical significance (which is common in meta-learning). Also, the current case study is limited to MLP networks for regression problems. Hence, more experiments in new case studies should be performed in order to have a better confidence concerning the proposed strategies. Other candidate algorithms and classes of learning problems can be adopted in new experiments.

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

References

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

[2] Angluin, D.: Queries and concept learning. Machine Learning **2** (1988) 319–342

[3] Bensusan, H., Alexandros, K.: Estimating the predictive accuracy of a classifier. In: 12th European Conf. on Machine Learning. (2001) 25–36

[4] Blum, A., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence **97**(1-2) (1997) 245–271

[5] Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Meta-Learning: Applications to Data Mining. Springer (2008)

[6] 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

[7] 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)

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

[9] Giraud-Carrier, C., Vilalta, R., Brazdil, P.: Introduction to the special issue on meta-learning. Machine Learning **54**(3) (2004) 187–193

[10] Huang, G.B., Wang, D., Lan, Y.: Extreme learning machines - a survey. International Journal of Machine Learning and Cybernetics **2**(2) (2011) 107–122

[11] Jankowski, N., Duch W., Grabczewski K.: Meta-learning in Computational Intelligence. Studies in Computational Intelligence **358**, 1st Edition, Springer (2011).

[12] Kalousis, A., Gama, J., Hilario, M.: On data and algorithms - understanding inductive performance. Machine Learning **54**(3) (2004) 275–312

[13] Koepf, C. In: Meta-Learning: Strategies, Implementations, and Evaluations for Algorithm Selection, Infix (2006)

[14] Knorr, E., Ng, R.: A unified notion of outliers: Properties and computation. In: Proceedings of the KDD. (1997)

[15] Leite, R., Brazdil, P.: Predicting relative performance of classifiers from samples. In: 22nd Inter. Conf. on Machine Learning. (2005)

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

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

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

[19] Melville, P., Mooney, R.: Diverse ensembles for active learning. In: Proceedings of the 21th International Conference on Machine Learning. (2004)

[20] Muslea, I., Minton, S., Knobrock, C.: Active learning with multiple views. Journal of Artif. Intel. Research **27** (2006) 203–233

[21] 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

[22] Prechelt, L.: A set of neural network benckmark problems and benchmarking rules. Tech. Report 21/94, Universitä Karlsruhe (1994)

[23] Prudêncio, R.B.C., Ludermir, T.B.: Meta-learning approaches to selecting time series models. Neurocomputing **61** (2004) 121–137

[24] Prudêncio, R.B.C., Ludermir, T.B.: Selective generation of training examples in active meta-learning. International Journal of Hybrid Intelligent Systems **5** (2008) 59–70

[25] Prudêncio, R.B.C., Ludermir, T.B.: Combining uncertainty sampling methods for active meta-learning. In: Proc. of the 9th International Conference on Intelligent Systems Design and Applications. (2009) 220–225

[26] Prudêncio, R.B.C., Ludermir, T.B.: Active Generation of Training Examples in Meta-Regression. In: Proc. of the International Conference on Artificial Neural Networks (ICANN). (2009) 30–39

[27] Prudêncio, R.B.C., Souto, M., Ludermir, T.B.: Selecting Machine Learning Algorithms Using the Ranking Meta-Learning Approach. In: Meta-learning in Computational Intelligence. N, Jankowski et al. (Eds.). Studies in Computational Intelligence **358**, 1st Edition, Springer (2011) 225-243.

[28] Vanschoren, J.: Meta-Learning Architectures - Collecting, Organizing and Exploiting Meta-Knowledge. In: Meta-learning in Computational Intelligence. N, Jankowski et al. (Eds.). Studies in Computational Intelligence **358**, 1st Edition, Springer (2011) 117-155.

[29] Raghavan, H., Madani, O., Jones, R.: Active learning with feedback on both features and instances. Pattern Recognition Letters **7** (2006) 1655–1686

[30] Riccardi, G., Hakkani-Tur, D.: Active learning - theory and applications to automatic speech recognition. IEEE Transactions on Speech and Audio Processing **13**(4) (2005) 504–511

[31] Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: Proc. 18th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA (2001) 441–448

[32] Sampaio, I., Ramalho, G., Corruble, V., Prudêncio, R.: 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

[33] Scheffer, T., Decomain, C.Wrobel, S.: Active hidden Markov models for information extraction. In: Proceedings of the International Conference on Advances in Intelligent Data Analysis. (2001) 309318

[34] Schohn, G., Cohn, D.: Less is more - active learning with support vector machines. In: Proceedings of the 17th International Conference on Machine Learning. (2000) 839–846

[35] Settles, B., Craven, M.: Seung, H.S., Opper, M., Sompolinsky, H.: An analysis of active learning strategies for sequence labeling tasks. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). (2008) 10691078

[36] Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: Computational Learning Theory. (1992) 287–294

[37] Small, K., Roth, D.: Margin-based active learning for structured predictions. In International Journal of Machine Learning and Cybernetics **1**(1) (2010) 3–25

[38] 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

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

[40] Soares, C.: Uci++, improved support for algorithm selection using datasetoids. Lecture Notes in Computer Science **5476** (2009) 499–506

[41] Soares, C., Brazdil, P., Kuba, P.: A meta-learning approach to select the kernel width in support vector regression. Machine Learning **54**(3) (2004) 195–209

[42] 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

[43] Teixeira, I.: Active cp: A method for speeding up user preferences acquisition in collaborative filtering systems. In: 16th Brazilian Symposium on Artificial Intelligence. (2002) 237–247

[44] Todorovski, L., Dzeroski, S.: Combining classifiers with meta decision trees. Machine Learning **50**(3) (2003) 223–249

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

[46] Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of Machine Learning Research **2** (2002) 45–66

[47] Wang, X., Dong, L.C., Yan, J.: Improving Generalization of Fuzzy IF–THEN Rules by Maximizing Fuzzy Entropy. IEEE Transactions on Fuzzy Systems **17**(3) (2009)

[48] Wang, X., Dong, C.R..: Maximum Ambiguity Based Sample Selection in Fuzzy Decision Tree Induction. IEEE Transactions on Knowledge and Data Engineering **PP**(99) (2011)