Active Meta-Learning with Uncertainty Sampling and Outlier Detection

Ricardo B. C. Prudêncio and Teresa B. Ludermir

***Abstract*— Meta-Learning has been used to predict the performance of learning algorithms based on descriptive features of the learning problems. Each training example in this context, i.e. each meta-example, stores the features of a given problem and information about the empirical performance obtained by the candidate algorithms on that problem. The process of constructing a set of meta-examples may be expensive, since for each problem avaliable for meta-example generation, it is necessary to perform an empirical evaluation of the candidate algorithms. Active Meta-Learning has been proposed to overcome this limitation by selecting only the most informative problems in the meta-example generation. In this work, we proposed an Active Meta-Learning method which combines Uncertainty Sampling and Outlier Detection techniques. Experiments were performed in a case study, yielding significant improvement in the Meta-Learning performance.**

I. Introduction

The selection of adequate algorithms for solving learning problems is an important aspect to the success of the Machine Learning process [1]. Meta-Learning is a framework developed in the field of supervised Machine Learning with the aim of automatically predicting algorithms performance, thus assisting users in the process of algorithm selection [2].

A training example in Meta-Learning (called *meta-example*) represents the experience obtained from empirically evaluating a set of candidate algorithms on a given learning problem. Specifically, each meta-example stores: (1) features of a given problem (e.g. number of training examples and number of attributes); and (2) performance information related to the algorithms when empirically evaluated to the problem. A *meta-learner* uses a set of such meta-examples to acquire knowledge relating algorithms performance to the features of the learning problems.

Generating a set of meta-examples may be a costly process, since in order to produce a single meta-example, it is necessary to perform an empirical evaluation of the candidate algorithms on a problem. Hence, the cost of generating a whole 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 amount of available problems.

In order to minimize the above difficulty, the use of Active Learning [3] has been proposed to support the generation of meta-examples [4], [5]. In Active Learning, the learner has some ability to decide at each moment which examples will be included in the training set. In the so-called *Active Meta-Learning* proposal, Active Learning techniques are used to reduce the set 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.

Ricardo B. C. Prudêncio - Department of Information Science, Federal University of Pernambuco, Av. dos Reitores s/n, 50670-901, Recife (PE), Brazil; email: prudencio.ricardo@gmail.com. Teresa B. Ludermir - Center of Informatics, Federal University of Pernambuco, PO Box: 7851, 50732-970, Recife (PE), Brazil; email: tbl@cin.ufpe.br.

In the current work, we propose an Active Meta-Learning approach which combines Uncertainty Sampling [6], a specific Active Learning method, and Outlier Detection techniques [7]. Uncertainty Sampling has already been applied in isolation to Active Meta-Learning, obtaining satisfactory results [4], [5]. However, the literature in Active Learning has pointed out that Uncertainty Sampling often fails by selecting examples which are outliers [8]. Hence, in our work we improved the performance of Uncertainty Sampling by removing outliers among the problems avaliable for meta-example generation.

The proposed hybrid method was evaluated in a meta-learning task which corresponds to predicting the pattern of performance of Multi-Layer Perceptron (MLP) networks for regression problems. Experiments performed on a set of 50 problems revealed a gain in the meta-learner performance by using the proposed method, compared to both a random procedure for selecting problems and the use of Uncertainty Sampling in isolation.

The remaining of this paper is organized as follows. Section II brings a brief presentation of Meta-Learning, followed by section III which describes some approaches for Active Learning. Section IV presents, in more details, the proposed solution and an implemented prototype. Section V presents the performed experiments and obtained results. Finally, section VI concludes the paper by presenting some future work.

II. Meta-Learning

According to [9], 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 empirical performance of learning algorithms to the features of the learning problems [2]. In this context, each *meta-example* is related to a learning problem and stores: (1) the features describing the problem (the *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 [10], [1]. Each meta-example commonly stores, as performance information, a class attribute which indicates the best algorithm for the problem (in terms of obtained accuracy), among a set of candidates [11], [12], [13]. 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 [14], 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 trees were induced to predict the applicability of the candidate algorithms.

In [1], the authors performed the labeling of meta-examples by deploying a clustering technique. 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 technique, aiming to identify groups (clusters) of problems in which the algorithms obtained specific patterns of performance (e.g. a cluster in which certain algorithms achieve 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.

Other Meta-Learning approaches have been proposed in the literature. For instance, the NOEMON system [15] combines a pool of meta-learners in order to provide rankings of candidate algorithms. In [16], [10], instance-based learning is used to provide rankings of algorithms taking into account both accuracy and execution time. In the Meta-Regression approach [17], [18], regression models are used to directly predict the numerical value of accuracy of the candidate algorithms.

III. Active Learning

Active Leaning is a paradigm of Machine Learning in which the learning algorithm has some control over the inputs on which it trains [3]. 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 [6], text classification [19] and information filtering [20].

Previous work in Active Learning has been concentrated in the *Selective Sampling* approach [6]. In this approach, the learning algorithm begins with a small training set of labeled examples and a potentially large set of unlabeled examples to select. At each moment, the learner selects the most informative unlabeled example and asks the teacher to annotate it. According to [21], the Selective Sampling methods can be distinguished on three main categories, *Uncertainty Sampling* methods, *Version Space Reduction* methods and *Error Reduction* methods, described below.

In Uncertainty Sampling methods [22], [6], [23], at each moment, the current learner is used to make predictions for the available unlabeled examples. Following, the method selects the unlabeled example for which the current learner has the highest uncertainty in its prediction. According to [21], these methods are straightforward and can be easily adapted for a great variety of Machine Learning algorithms.

The Version Space Reduction methods (or committee-based methods) [24], [19], [25] deploy an idea which is similar to Uncertainty Sampling. Here, 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 each unlabeled example. A high degree of disagreement on the predictions of the committee is defined as a measure of uncertainty.

In the Error Reduction methods [8], [6], the selected example is the one that minimizes the expected error of the learner, once labeled and included in the training set. Although more sophisticated, they are computationally expensive, since for each unlabeled example and possible label, it is necessary to re-train the learner in order to compute the expected reduction in the error rate [8].

IV. Active Meta-Learning

A limitation that can be pointed out in the process of Meta-Learning is related to the generation of meta-examples. Given a learning problem, in order to produce a meta-example, it is necessary to perform an empirical evaluation of the available algorithms in order to collect its performance information on the problem. Although the proposal of Meta-Learning is to perform this empirical evaluation only in a limited number of problems, the cost of generating a whole set of meta-examples may be high, depending, for instance, on the number and complexity of the candidate algorithms and the amount of data available in the learning problems.

In order to minimize the above difficulty, in [4], [5] the authors proposed the Active Meta-Learning, in which the generation of meta-examples is supported by using Active Learning techniques. In Active Meta-Learning, the use of Active Learning techniques improves the efficiency of 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 represents the process of generating meta-examples by following our proposal. Initially, the meta-features are computed for each available problem, in order to generate a set of *unlabeled* meta-examples. Each unlabeled meta-example stores the description of a problem, but the performance information of the candidate algorithms is not known yet.

In order to generate *labeled* meta-examples, the Active Learning module selects those unlabeled meta-examples considered the most relevant for the Meta-Learning task. The selection of unlabeled meta-examples is performed based on a pre-defined Active Learning method implemented in the module.

Given the selected unlabeled meta-example, the candidate algorithms are then empirically evaluated on the related problem, in order to collect the performance information. Each new labeled meta-example (composed by meta-features and performance information) is then stored in the training set of the Meta-Learner module. This module in turn will use this training set to acquire knowledge relating meta-features to the performance of the candidate algorithms.

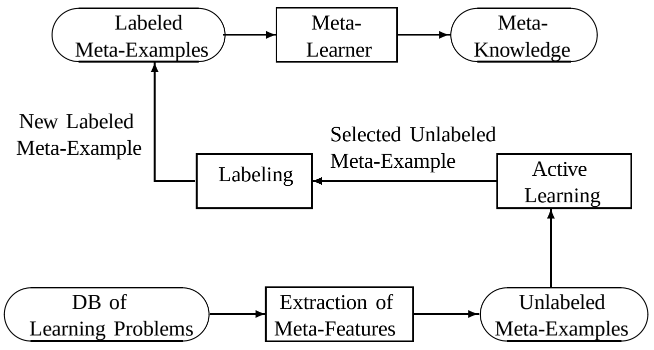

Fig. 1. Active generation of meta-examples

In [4], [5], an Active method based on *Uncertainty Sampling* [6] was used to select meta-examples for a k-NN (k-Nearest Neighbors) algorithm used as meta-learner. The experiments performed in [4], [5] have shown a significant gain in meta-learning performance when the Uncertainty Sampling method is used. A limitation of Uncertainty Sampling, however, is that it often selects examples that are outliers [8]. Such examples have in fact a high degree of uncertainty but they should not be considered as informative.

In the current work, we extend our previous research by combining Uncertainty Sampling and Outlier Detection techniques, applied to Active Meta-Learning. In this combination, we first removed unlabeled meta-examples considered as outliers, and then applied an Uncertainty Sampling technique in order to progressively select from the remaining unlabeled meta-examples the most informative ones to be labeled.

In the next subsections, we present details about an implemented prototype which followed the above proposal. Section V brings the experiments performed in a case study by using the implemented prototype.

A. 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 algorithms performance). In the implemented prototype, we used the k-NN algorithm which has some advantages when applied to Meta-Learning [10]. For instance, when a new meta-example becomes available, it can be easily integrated without the need to initiate re-learning [10]. In this section, we provide a description of the meta-learner based on the k-NN algorithm.

Let $E = \{e_1, \ldots, e_n\}$ be the set of n problems already used to generate a set of n labeled meta-examples $ME = \{me_1, \ldots, me_n\}$. Each meta-example is related to a problem and stores the values of p features $X_1, \ldots, X_p$ 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:

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

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.

B. 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 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.

1) Outlier Detection: In our prototype, we adapted the Distance-Based method proposed in [7] 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:

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

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:

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

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.

2) Uncertainty Sampling: As said, in this Active Learning method, the learner uses the currently labeled examples to generate a prediction for each unlabeled example. A degree of uncertainty of the provided prediction is assigned for each unlabeled example. Finally, the active method selects the example with highest uncertainty.

The classification uncertainty of the k-NN algorithm is defined in [6] 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.

Formally, let ME be the set of labeled meta-examples, and let $\widetilde{ME}$ be the set of unlabeled meta-examples. Let ME_l be the subset of labeled meta-examples associated to the class label c_l, i.e. $ME_l = \{me_i \in ME | C(e_i) = c_l\}$. Given the set ME, the classification uncertainty of k-NN for each $\widetilde{me} \in \widetilde{ME}$ is defined as:

$$\mathcal{S}(\widetilde{me}|ME) = \frac{\min_{me_i \in ME} dist(\widetilde{\mathbf{x}}, \mathbf{x}_i)}{\sum_{l=1}^{L} \min_{me_i \in ME_l} dist(\widetilde{\mathbf{x}}, \mathbf{x}_i)} \qquad (4)$$

In the above equation, $\widetilde{\mathbf{x}}$ is the problem description stored in $\widetilde{me}$. The AL module then selects, to be labeled, the unlabeled meta-example $\widetilde{me}^* \in \widetilde{ME}$ with highest uncertainty:

$$\widetilde{me}^* = argmax_{\widetilde{me} \in \widetilde{ME}} \mathcal{S}(\widetilde{me}|ME) \qquad (5)$$

Finally, the selected meta-example 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 $\widetilde{e}^*$.

V. Case Study

In this section, we present the application of the implemented prototype in a case studies that correspond to a meta-learning task originally presented in [4]. Details about this task will be provided in this section, followed by the performed experiments and obtained results.

The meta-learning task consists in predicting a class label related to the performance of Multi-Layer Perceptron (MLP) networks for regression problems. The set of meta-examples in this case study was generated from the application of MLP to 50 regression problems, available in the WEKA project[1]. Hence, 50 different meta-examples were generated.

Each meta-example in this case study stored the values of $p = 10$ meta-features, which correspond to:

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 [4], each meta-example also stored the value of a class attribute which indicated the performance pattern obtained by the MLP network when applied to the problem. More specifically, each meta-example was assigned to one of the class labels: *cluster1*, corresponding to problems in which the MLP obtained good test error rates; and *cluster2*, corresponding to tasks in which the MLP obtained from low to medium test error rates. These class labels were defined after an empirical evaluation (using a cross validation experiment) of the MLP on the 50 regression problems, and a cluster analysis of the obtained results.

A. Experiments Description

In order to evaluated the performance of the k-NN meta-learner by using the Uncertainty Sampling with Outlier Detection, we performed a leave-one-out experiment, which is described just below.

[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/

At each step of leave-one-out, one meta-example is left out for testing the Meta-Learner, and the remaining 49 meta-examples are considered as candidates to be included in the training set. Initially, 5 candidate meta-examples (about 10% of the meta-examples) were 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. At each included meta-example, the Meta-Learner is judged on the test problem left out, receiving either 1 or 0 for failure or success. Hence, a curve with 44 binary judgments is produced for each test problem. Finally, a curve of 44 error rates obtained by Meta-Learner can be computed by averaging the curves of judgments over the 50 steps of the leave-one-out experiment.

In our work, we also performed experiments with the Uncertainty Sampling without using Outlier Detection, which corresponds to the Active method evaluated in our previous research [4], [5]. The motivation here is to evaluate the usefulness of Outlier Detection to improve the Uncertainty Sampling method. We followed the same methodology of experiments as described above, however at each step of leave-one-out, all the remaining 49 meta-examples were incrementally included in the training set. Hence, in this experiment, a curve of 49 error rates was generated.

As a basis of comparison, the leave-one-out experiment was applied by using Random Sampling for selecting unlabeled problems. According to [6], despite its simplicity, the random method has the advantage of performing a uniform exploration of the example space.

Finally, we highlight that each of the three above methods were evaluated for different configurations of the k-NN meta-learner (with k = 1, 3, 5, 7, 9 and 11 nearest neighbors). For each configuration, 30 runs of experiments were executed.

B. Results

Figure 2 presents the curve of error rates obtained by the k-NN meta-learner averaged across the different configurations of the parameter k and runs of experiments. As it was expected, the Uncertainty Sampling was better than the Random Sampling in both executions: with and without Outlier Detection. This result indicates the viability of using Active Learning methods for selecting meta-examples.

The use of Outlier Detection steadily improved the performance of the Uncertainty Sampling from 14 to 39 meta-examples included in the training set, which correspond to 26 points in the curve of error rates. By applying a t-test (95% of confidence) to the difference of error rates, we observed that the performance gain obtained with the use of Outlier Detection was statistically significant in 16 points in the curve of error rates.

Figure 3 presents the average gain in performance obtained by the Uncertainty Sampling (with and without Outlier Detection), relative to the Random Sampling for different intervals of the number of meta-examples in the training set. For each considered interval, we observed an improvement in performance by using Outlier Detection. However this improvement was higher as the number of meta-examples in the training set increased, which indicates that, without Outlier Detection, the inclusion of outliers in the training set progressively harmed the performance of the meta-learner.

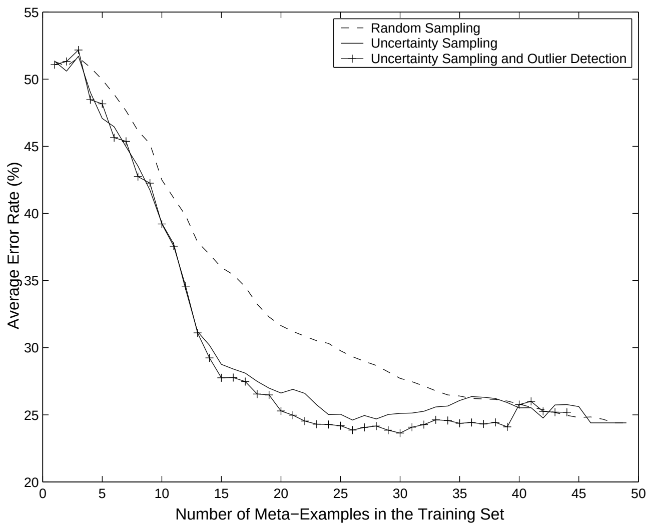

Fig. 2. Average curves of error rates for the Random Sampling and the Uncertainty Sampling (without and with Outlier Detection)

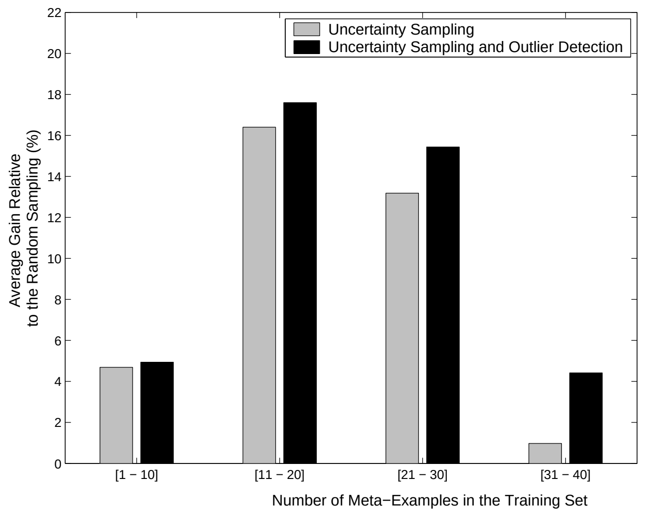

Fig. 3. Average gain relative to the Random Sampling for different intervals of the number of meta-examples in the training set

VI. Conclusion

In this paper, we presented the use of an Active Meta-Learning method which combined Uncertainty Sampling and Outlier Detection techniques. The Uncertainty Sampling was used to support the selection on informative examples for Meta-Learning. In order to improve the Uncertainty Sampling, an Outlier Detection technique was used to remove from the set of learning problems those ones considered as outliers for meta-example generation.

An implemented prototype which used a k-NN meta-learner was evaluated in a case study. In the performed experiments, the performance obtained by using the proposed method was better than the results obtained in previous work on Active Meta-Learning. The experiments revealed that the Uncertainty Sampling method was in fact sensitive to the presence of outliers, in such a way that its performance was improved when the Outlier Detection technique was applied.

Finally, we highlight some aspects of our work which will be investigated in the future. First, other Outlier Detection techniques can be applied to improve the Uncertainty Sampling. In future work, we intend to evaluate the performance of the proposed method compared to other Active methods (e.g. error-reduction methods). Finally, we also intend to developed Active methods for other Meta-Learning techniques.

VII. Acknowledgments

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

References

[1] A. Kalousis, J. Gama, and M. Hilario, "On data and algorithms - understanding inductive performance," *Machine Learning*, vol. 54, no. 3, pp. 275–312, 2004.

[2] C. Giraud-Carrier, R. Vilalta, and P. Brazdil, "Introduction to the special issue on meta-learning," *Machine Learning*, vol. 54, no. 3, pp. 187–193, 2004.

[3] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, pp. 201–221, 1994.

[4] R. B. C. Prudêncio and T. B. Ludermir, "Active learning to support the generation of meta-examples," in *Proc. of the International Conference on Artificial Neural Networks*, 2007, pp. 817–826.

[5] ——, "Active selection of training examples for meta-learning," in *Proc. of the International Conference on Hybrid Intelligent Systems*, 2007.

[6] M. Lindenbaum, S. Markovitch, and D. Rusakov, "Selective sampling for nearest neighbor classifiers," *Machine Learning*, vol. 54, pp. 125–152, 2004.

[7] E. Knorr and R. Ng, "A unified notion of outliers: Properties and computation," in *Proceedings of the KDD*, 1997.

[8] N. Roy and A. McCallum, "Toward optimal active learning through sampling estimation of error reduction," in *Proc. 18th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2001, pp. 441–448.

[9] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Journal of Artificial Intelligence Review*, vol. 18, no. 2, pp. 77–95, 2002.

[10] P. Brazdil, C. Soares, and J. da Costa, "Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results," *Machine Learning*, vol. 50, no. 3, pp. 251–277, 2003.

[11] 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*, vol. 25, no. 8, pp. 911–921, 2004.

[12] R. B. C. Prudêncio and T. B. Ludermir, "Meta-learning approaches to selecting time series models," *Neurocomputing*, vol. 61, pp. 121–137, 2004.

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

[14] D. J. S. D. Michie and C. C. Taylor, Eds., *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, New York, 1994.

[15] A. Kalousis and T. Theoharis, "Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection," *Intelligent Data Analysis*, vol. 3, no. 5, pp. 319–337, 1999.

[16] C. Soares and P. Brazdil, "Zoomed ranking - selection of classification algorithms based on relevant performance information," *Lecture Notes in Computer Science*, vol. 1910, pp. 126–135, 2000.

[17] C. Koepf, C. 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.

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

[19] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of Machine Learning Research*, vol. 2, pp. 45–66, 2002.

[20] I. Sampaio, G. Ramalho, V. Corruble, and R. 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, pp. 107–110.

[21] I. Muslea, S. Minton, and C. Knobrock, "Active learning with multiple views," *Journal of Artificial Intelligence Research*, vol. 27, pp. 203–233, 2006.

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

[23] H. Raghavan, O. Madani, and R. Jones, "Active learning with feedback on both features and instances," *Pattern Recognition Letters*, vol. 7, pp. 1655–1686, 2006.

[24] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Computational Learning Theory*, 1992, pp. 287–294.

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