Active Selection of Training Examples for Meta-Learning

Ricardo B. C. Prudêncio
Department of Information Science
Federal University of Pernambuco
Av. dos Reitores, s/n - CEP 50670-901 - Recife (PE) - Brazil
prudencio.ricardo@gmail.com

Teresa B. Ludermir
Center of Informatics
Federal University of Pernambuco
Pobox 7851 - CEP 50732-970 - Recife (PE) - Brazil
tbl@cin.ufpe.br

Abstract

Meta-Learning has been used to relate the performance of algorithms and the features of the problems being tackled. The knowledge in Meta-Learning is acquired from a set of meta-examples which are generated from the empirical evaluation of the algorithms on problems in the past. In this work, Active Learning is used to reduce the number of meta-examples needed for Meta-Learning. The motivation is to select only the most relevant problems for meta-example generation, and consequently to reduce the number of empirical evaluations of the candidate algorithms. Experiments were performed in two different case studies, yielding promissing results.

1 Introduction

One of the major challenges in several domains of application is to predict when one algorithm is more adequate than another to solve a particular problem [9]. 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 [7, 23].

The knowledge in Meta-Learning is acquired from a set of training examples (the *meta-examples*) that store the experience obtained in the application of a number of candidate algorithms in problems investigated in the past. More specifically, each meta-example is related to a given problem and stores: (1) the features that describe the problem; and (2) information about the performance obtained by the algorithms when applied to the problem.

A limitation of Meta-Learning is related to the process of generating meta-examples. In order to generate a meta-example from a given problem, it is necessary to perform an empirical evaluation (e.g. cross-validation) to collect the performance information of the algorithms. 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 data available in the problems.

In this paper, we present the use of Active Learning [5] to support the generation of meta-examples. The main motivation of Active Learning is to reduce the number of training examples, at same time maintaining the performance of the learning algorithms. In our proposal, it corresponds to reduce the set of meta-examples, consequently, reducing the number of empirical evaluations performed on the candidate algorithms.

In [17], we presented the initial experiments performed to evaluate the viability of the proposed solution. In that work, an Active method based on *Classification Uncertainty* [14] was used to select meta-examples for a k-NN (k-Nearest Neighbors) algorithm used as meta-learner. In the current work, we present new experiments that evaluated the proposed solution, which was applied to two different case studies. Experiments revealed a gain in the meta-learner performance by using the Active Learning method.

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

2 Meta-Learning

Meta-Learning is a framework that defines techniques to assist algorithm selection for learning problems (usually classification and regression problems) [7]. Each training example (or *meta-example*) is related to an individual problem investigated in the past and stores: (1) a set of features (called *meta-attributes*) that describes the problem; and (2) the performance information, derived from the empirical evaluation of the candidate algorithms on the problem.

The meta-attributes usually are statistical and information theory measures of the problem's dataset, such as number of training examples and attributes, correlation between attributes, class entropy, presence of outliers, among others [3, 9]. In a strict formulation of Meta-Learning, the performance information is a class attribute which indicates the best algorithm for the problem, among a set of candidate algorithms. The class label stored in a meta-example is usually defined via a cross-validation experiment using the available problem's dataset. The *meta-learner* in this case is simply a classifier which predicts the best algorithm for a given problem based on its descriptive meta-attributes [1].

Although the strict Meta-Learning has been investigated by different authors (see for instance [1, 9, 12, 15, 16, 18]), other Meta-Learning techniques have been proposed to provide more informative solutions to algorithm selection. In [6], the authors proposed a meta-learner not only to predict the best algorithm but also to predict the applicability of each candidate algorithm to the new problems being tackled. In [10], the NOEMON system combined different strict meta-learners in order to provide rankings of the candidate algorithms. In [3], the authors applied instance-based learning to provide rankings of algorithms, taking into account the predicted accuracy and execution time of the algorithms. In [2], the authors used a regression model as meta-learner in order to predict the numerical value of the accuracy for each candidate algorithm.

3 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 [5]. The main objective of this paradigm is to reduce the number of training examples, at same time maintaining 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 [14], text classification [22] and information filtering [20].

Previous work in Active Learning has been concentrated in the *selective sampling* approach [14]. 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.

In *certainty-based* methods [13] for selective sampling, 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 *committee-based* methods [21] deploy a similar idea, however the predictions are generated by a committee of learners, instead of a single learner. In this case, a high degree of disagreement on the predictions indicates that an unlabeled example is informative. In the *direct* methods [19], the selected example is the one that minimizes the expected error of the learner, once labeled and included in the training set.

4 Active Learning for Meta-Example Generation

As seen, in order to generate a meta-example, it is necessary to perform an empirical evaluation of the candidate algorithms on a given problem. The generation of a set of meta-examples may be a costly process depending for instance on the methodology of empirical evaluation, the number of available problems, and the number and complexity of the candidate algorithms. In this context, the use of Active Learning may improve the Meta-Learning process by reducing the number of required meta-examples, and consequently the number of empirical evaluations on the candidate algorithms.

Figure 1 presents the architecture of system following our proposal, which has three phases. In the meta-example generation phase, the Active Learning (AL) module selects from a base of problems, the most informative for the Meta-Learning task. The candidate algorithms are then evaluated on the selected problems, in order to generated a new meta-example. In the training phase, the Meta-Learner (ML) acquires knowledge from the generated meta-examples, associating meta-attributes of the problems to the performance of the algorithms. Finally, in the use phase, given an input problem, the Feature Extractor (FE) module extracts the values of the meta-attributes, and according to the knowledge acquired in the training phase, the ML module predicts the performance information of the algorithms.

In order to evaluate the proposal, we implemented a prototype which was applied in two different case studies. In this prototype, the k-Nearest Neighbors (k-NN) algorithm was used in the ML module, and an Active Learning method based on classification uncertainty of the k-NN [14] is used in the AL module. In the next sections, we provide more details of the proposed implemented prototype. In section 5, we present the two case studies as well as the experiments

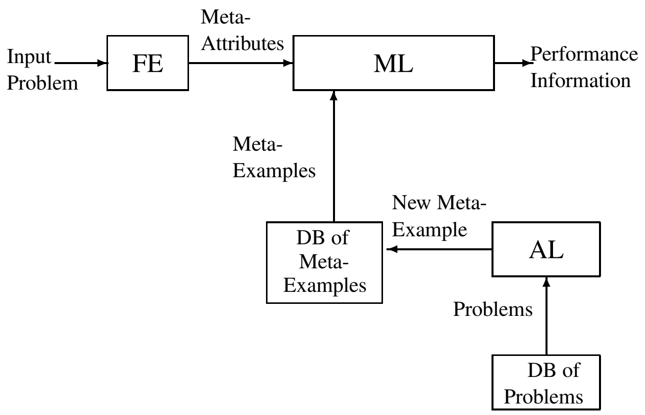

Figure 1. System Architecture.

and obtained results.

4.1 Meta-Learner

The Meta-Learner in the prototype corresponds to a conventional classifier, and it is applicable to tasks in which the performance information is formulated as a class attribute (e.g. the class associated to the best algorithm or the class related to patterns of algorithms performance). In the implemented prototype, we used the k-NN algorithm which has some advantages when applied to Meta-Learning [3]. For instance, when a new meta-example becomes available, it can be easily integrated without the need to initiate relearning [3]. 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 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 $D = \{c_1, \ldots, c_L\}$ 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 $c_l \in D$.

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 $c_l \in D$ in the class labels associated to the retrieved meta-examples.

4.2 Active Learning

The ML module acquires knowledge from a set of meta-examples, which correspond to *labeled* problems. The AL module receives a set of *unlabeled* problems, i.e. the problems in which the candidate algorithms were not yet evaluated. The AL module incrementally selects unlabeled problems to be used for generating new meta-examples.

In the prototype, the AL module implements a certainty-based method (see section 3) which selects the unlabeled example for which the current learner has the highest uncertainty in its prediction. The classification uncertainty of the k-NN algorithm is defined in [14] as the ratio of: (1) the distance between the unlabeled example and its nearest labeled neighbor; and (2) the sum of the distances between the unlabeled example and its nearest labeled neighbors of different classes.

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

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

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

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

$$\widetilde{e}^* = argmax_{\widetilde{e} \in \widetilde{E}} \mathcal{S}(\widetilde{e}|E) \qquad (3)$$

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

5 Case Studies

In this section, we present the application of the implemented prototype to two different case studies that correspond to two meta-learning tasks originally presented in previous work [16, 8]. Each case study provides a set of meta-examples which was used in the current work to perform experiments to evaluate the implemented prototype.

5.1 Case Study I

In the first case study, the implemented prototype was evaluated in a meta-learning task originally proposed in [15] which consisted in selecting between two candidate algorithms for time series forecasting problems: the Time-Delay Neural Network (TDNN) [11] and the Simple Exponential Smoothing model (SES) [4]. In [15], a set of meta-examples was generated from the evaluation of TDNN and SES on 99 time series collected from the Time Series Data Library [1]. Hence, 99 meta-examples were generated.

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

1. Length of the time series (X_1);
2. Mean of the absolute values of the 5 first autocorrelations (X_2);
3. Test of significant autocorrelations (X_3);
4. Significance of the first, second and third autocorrelation (X_4, X_5 and X_6);
5. Coefficient of variation (X_7);
6. Absolute value of the skewness and kurtosis coefficient (X_8 and X_9);
7. Test of Turning Points for randomness (X_{10}).

In this case study, the labeling of a time series (i.e. definition of the class attribute for training meta-examples) is performed through the empirical evaluation of TDNN and SES in forecasting the series. For this, a hold-out experiment was performed, as described in [15]. Given a time series, its data was divided into two parts: the fit period and the test period. The test period consists on the last 30 points of the time series and the fit period consists on the remaining data. The fit data was used to calibrate the parameters of both models TDNN and SES. Both calibrated models were used to generate one-step-ahead forecasts for the test data. Finally, the class attribute was assigned as the model which obtained the lowest mean absolute forecasting error on the test data.

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

5.1.1 Experiments

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

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

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 method for selecting unlabeled problems. According to [14], despite its simplicity, the random method has the advantage of performing a uniform exploration of the example space.

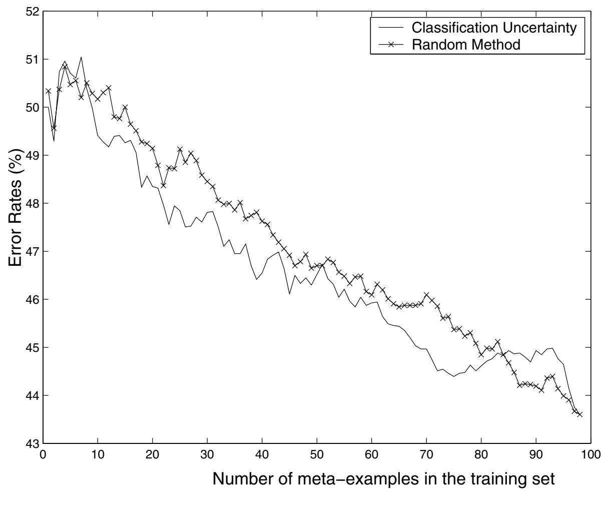

Figure 2. Case Study I - Average curves of error rates for both the Classification Uncertainty and the Random method.

5.1.2 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. The figure presents the average curve obtained when both methods were used: the Classification Uncertainty (described in section 3.3) and the Random method. As it is expected, for both methods, the error rate obtained by the ML module decreased as the number of meta-examples in the training set increased. However, the error rates obtained by deploying the Classification Uncertainty method were, in general, lower than the error rates obtained by deploying the Random method. In fact, from 8 to 84 meta-examples included in the training set, the Classification Uncertainty method steadily achieved better performance compared to the Random method.

Despite the performance gain obtained by Classification Uncertainty in absolute terms, the statistical difference compared to the Random method was not so significant. By applying a t-test (95% of confidence) to the difference of error rates, we observed that the Classification Uncertainty obtained a statistical gain in 10 points of the curve of error rates, which represents only about 10% of the 98 points.

5.2 Case Study II

In the second case study, the prototype was evaluated in a meta-learning task proposed in [8] which consisted in predicting the performance pattern of Multi-Layer Perceptron (MLP) networks for regression problems. Below, we provide a brief description of the meta-examples related to this task. More details can be found in [8].

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

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

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

In [8], each meta-example was assigned to one 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 tasks, and a cluster analysis of the obtained results.

5.2.1 Experiments

The experiments performed on this case study followed the same methodology applied in the first case study. The ML module was evaluated for different values of the parameter k (1, 3, 5, 7, 9 and 11). As in the first case study, the ML module was evaluated by progressively including meta-examples in its training set. The methodology of experiments was applied for both the Classification Uncertainty and the Random procedures used in the AL module and the average curves of error rates were computed.

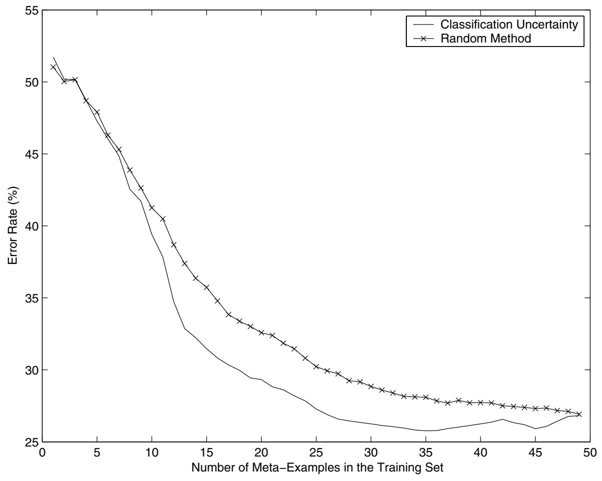

Figure 3. Case Study II - Average curves of error rates for both the Classification Uncertainty and the Random method.

5.2.2 Results

As in the first case study, the error rates decreased as the number of meta-examples in the training set increased, considering both the Classification Uncertainty and the Random method. However the curves of error rates in the second case study were more regular, showing a lower degree of oscillation in the error rates (see figure 3). In absolute terms, the results obtained by the Classification Uncertainty were better than the Random method in the most part of the curve of error rates, more specifically from 5 to 48 meta-examples in the training set.

The good results of the classification uncertainty were also observed to be statistically significant. A t-test (95% of confidence) applied to the difference of error rates indicated that the classification uncertainty obtaining a gain in performance in 30 points in the curve of error rates (about 61% of the points).

6 Conclusion

In this paper, we presented the use of Active Learning to support the selection on informative examples for Meta-Learning. A prototype was implemented using the k-NN algorithm as meta-learner and a certainty-based method for Active Learning. The prototype was evaluated in two different case studies, and the results obtained by the Active Learning method were in general better than a Random method for selecting meta-examples.

We can point out contributions of our work to two different fields: (1) in the Meta-Learning field, we proposed a solution to speed up the construction of a good set of examples for Meta-Learning; and (2) in the Active Learning field, we applied its concepts and techniques in a context which had not yet been investigated.

The current work still have limitations which will be dealt with in future work. First, we only deploy a specific certainty-based method for Active Learning. In future work, we intend to evaluate the performance of other Active Learning methods (e.g. committee-based methods) in the context of Meta-Learning. We also intend to investigate the use of Active Learning for other Meta-Learning techniques (as those cited in section 2).

References

[1] D. Aha. Generalizing from case studies: A case study. In *Proceedings of the 9th International Workshop on Machine Learning*, pages 1–10. Morgan Kaufmann, 1992.

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

[3] P. Brazdil, C. Soares, and J. da Costa. Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning*, 50(3):251–277, 2003.

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

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

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

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

[8] S. B. Guerra, R. B. C. Prudêncio, and T. B. Ludermir. Meta-aprendizado de algoritmos de treinamento para redes multi-layer perceptron. In *Anais do VI Encontro Nacional de Inteligência Artificial*, pages 1022–1031, 2007.

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

[10] A. Kalousis and T. Theoharis. Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis*, 3(5):319–337, 1999.

[11] K. J. Lang and G. E. Hinton. A time-delay neural network architecture for speech recognition. Technical Report CMU-DS-88-152, Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Dec. 1988.

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

[13] 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*, pages 3–12, 1994.

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

[15] R. B. C. Prudêncio and T. B. Ludermir. Selection of models for time series prediction via meta-learning. In *Proceedings of the Second International Conference on Hybrid Systems*, pages 74–83. IOS Press, 2002.

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

[17] 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*, page (to appear), 2007.

[18] R. B. C. Prudêncio, T. B. Ludermir, and F. A. T. de Carvalho. A modal symbolic classifier to select time series models. *Pattern Recognition Letters*, 25(8):911–921, 2004.

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

[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*, pages 107–110, 2006.

[21] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Computational Learning Theory*, pages 287–294, 1992.

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

[23] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Journal of Artificial Intelligence Review*, 18(2):77–95, 2002.