Active Generation of Training Examples in Meta-Regression

Ricardo B.C. Prudêncio and Teresa B. Ludermir

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

Abstract. Meta-Learning predicts the performance of learning algorithms based on features of the learning problems. Meta-Learning acquires knowledge from a set of meta-examples, which store the experience obtained from applying the algorithms to problems in the past. A limitation of Meta-Learning is related to the generation of meta-examples. In order to construct a meta-example, it is necessary to empirically evaluate the algorithms on a given problem. Hence, the generation of a set of meta-examples may be costly depending on the context. In order to minimize this limitation, the use of Active Learning is proposed to reduce the number of required meta-examples. In this paper, we evaluate this proposal on a promising Meta-Learning approach, called Meta-Regression. Experiments were performed in a case study to predict the performance of learning algorithms for MLP networks. A significant performance gain was observed in the case study when Active Learning was used to support the generation of meta-examples.

1 Introduction

Meta-Learning is a framework developed in the field of supervised machine learning with the aim of relating features of the learning problems to the performance of the learning algorithms [1]. The knowledge in Meta-Learning is acquired from a set of *meta-examples*, in which each meta-example stores the experience obtained from the application of a set of candidate algorithms in a particular learning problem. Meta-Learning automatically captures the expertise gained on different problems, which will be used to predict learning performance [2].

Different Meta-Learning approaches have been proposed [1]. An approach that revealed promissing results in the literature is the Meta-Regression [3,4,5]. In this approach, each meta-example is related to a problem and stores: (1) features of the problem (e.g., number of training examples and attributes); and (2) the performance information (e.g., classification accuracy) estimated from the empirical evaluation of a set of candidate algorithms on the problem. In Meta-Regression, a regression algorithm is used to predict the performance information based on the features of the problems.

Generating a set of training examples for Meta-Regression 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 [6] is proposed to support the generation of meta-examples for Meta-Regression. Active Learning is mainly focused on domains of application in which it is hard to produce training examples, which is the case of Meta-Regression. In our context, Active Learning techniques are used to select the most relevant problems for generating meta-examples, consequently reducing the effort in performing empirical evaluations on the candidate algorithms.

In our work, a prototype was implemented in order to evaluate the proposed approach. In this prototype, the Linear Regression (LR) algorithm was used as meta-regressor to predict the performance difference between two algorithms for training Multi-Layer Perceptron (MLP) neural networks [7]. An active technique based on uncertainty of prediction was applied in the selection of problems for meta-example generation. Experiments were performed on a set of 50 problems, comparing the active technique to a random (passive) strategy for selecting problems. The performed experiments revealed a performance gain in the meta-regressor when the active learning technique was considered.

Section 2 brings a brief presentation of Meta-Learning, followed by section 3 which describes the proposed solution and the implemented prototype. Section 4 presents the performed experiments and obtained results. Finally, section 5 concludes the paper by presenting some final considerations and future work.

2 Meta-Learning

Meta-Learning is focused in our work as the automatic process of acquiring knowledge that relates the performance of learning algorithms to the features of the learning problems [1]. In this context, each *meta-example* is related to a learning problem and stores: (1) the features describing the problem, called *meta-features*; and (2) information about the performance of one or more algorithms when applied to the problem. The *meta-learner* is a learning system that receives as input a set of such meta-examples and then acquires knowledge used to predict the algorithms performance for new problems being solved.

The meta-features are, in general, statistics describing the training dataset of the problem, such as number of training examples, number of attributes, correlation between attributes, class entropy, among others [8,9]. In a strict formulation of Meta-Learning, each meta-example stores, as performance information, a class label which indicates the best algorithm for the problem, among a set of candidates [10]. 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.

Although the strict Meta-Learning approach (as described above) has been applied by different authors (such as [11,12,13]), certain information loss may be introduced in the definition of the class labels associated to meta-examples. For instance, the performance of two algorithms may be very similar, and this information will be lost by merely recording the best algorithm as class label [5].

In order to overcome the above difficulty, the Meta-Regression approach [3,5,4] tries to directly predict a numerical performance measure (e.g., accuracy, error difference, rank,...) of the candidate algorithms. In this case, the meta-examples store as performance information the numerical performance obtained in previous problems. The meta-learner, in turn, is a regression model that may be used either to select the best candidate algorithm based on the highest predicted performance or to provide a ranking of algorithms based on the order of predicted performances.

In [3], the authors evaluated different algorithms as meta-regressors, including linear regression models, piecewise linear models, decision trees and instance-based regression. In [4], the authors used linear regression models to predict the accuracy of 8 classification algorithms, and the experiments revealed good results. In [5], the authors performed comparative experiments with both the strict Meta-Learning and Meta-Regression approaches, and observed that the latter one performed better when used to support algorithm selection.

3 Active Generation of Meta-examples

As said in section 2, Meta-Learning accumulates the experience stored in meta-examples which are derived from the empirical evaluation of the candidate algorithms on problems solved in the past. The generation of a set of meta-examples may be a costly process depending, for instance, on the complexity of the candidate algorithms, the cost of the methodology used for empirical evaluation, and the number and size of the available problems.

In order to minimize the above difficulty, Active Learning [6] was proposed in [14] in order to select the most relevant problems for meta-example generation, aiming to reduce the effort in performing empirical evaluations with the candidate algorithms. Active Learning is a paradigm in which the learning algorithm has some control over the inputs on which it trains [6]. Active Learning is ideal for domains in which the acquisition of labeled examples is a costly process. The use of Active Learning techniques in our context aims to reduce the number of meta-examples, and consequently the number of empirical evaluations of the algorithms, at same time maintaining the performance of the meta-learner.

Figure 1 represents the proposed approach. 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

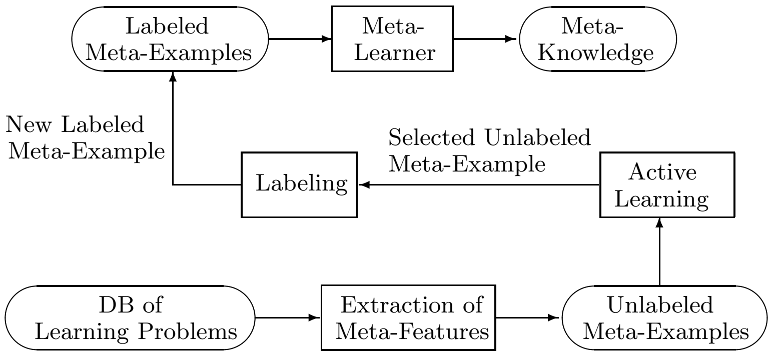

Fig. 1. Active generation of meta-examples

information. Each new labeled meta-example (composed of 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.

The viability of this proposal was verified in [14] through a number of experiments in which an uncertainty sampling method improved the performance of a strict meta-learner. In this paper, we extend our previous work, by evaluating the use of Active Learning for Meta-Regression. As said, Meta-Regression is more flexible when compared to the strict Meta-Learning, and it has been a topic of interest for different authors, which motivated the current work. We highlight that the active techniques evaluated in our previous work [14] are not suitable to Meta-Regression, and hence, new techniques were investigated in the current research.

In order to evaluate the *Active Meta-Regression*, a prototype was implemented and applied to a meta-learning task which corresponds to predict the difference in performance between the Backpropagation (BP) [7] and the Levenberg-Marquardt (LM) [15] algorithms used to train MLP neural networks. The prediction of this performance information supports the choice between BP and LM to new problems at hand. In this task, each meta-example stored: (1) the value of 10 descriptive meta-features of a regression problem; and (2) the error difference between BP and LM observed in the problem. In this prototype, the Linear Regression (LR) algorithm was used as meta-regressor to predict the performance information based on the meta-features. An active method based on uncertainty of prediction was used to select meta-examples for the LR algorithm.

In the next subsections, we present details about the implemented prototype such as the definition of the meta-examples (section 3.1), the description of the meta-regressor (section 3.2) and the description of the Active Learning method (section 3.3).

3.1 Meta-examples

The set of meta-examples is generated from the application of the candidate algorithms on a set of learning problems. Formally, 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 single problem and stores the values of p features $X_1, \ldots, X_p$ for the problem and the value of a target attribute Y. In this way, each meta-example $me_i \in ME$ is represented as the pair (x_i, y_i) storing: (1) the description x_i of the problem e_i, where $x_i = (x_i^1, \ldots, x_i^p)$ and $x_i^j = X_j(e_i)$; and (2) the performance information y_i estimated for e_i, where $y_i = Y(e_i)$.

Meta-Features. The first step to generate a meta-example from a problem is to extract its meta-features. In our work, a total number of $p = 10$ meta-features adopted in [14] 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.

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.

Performance Information. The second step to generate a meta-example is to estimate the performance of the candidate algorithms on the problem being tackled. In our prototype, this step consists of evaluating the performance of BP and LM when used to train one-hidden layer MLPs[1]. From this evaluation, we produce the performance information (Y) stored in the meta-examples which will correspond to the difference between the estimated performance of BP and LM. In order to measure the performance of each training algorithm in each problem, the following methodology of evaluation was applied.

The 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 [16]: early stopping was used to avoid overfitting with the GL_5

[1] The BP and LM algorithms were implemented by using the NNET Matlab toolbox. Learning rates were defined by default.

stopping criterion and a maximum number of 1000 training epochs (see [16] 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:

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

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 $\overline{t}$ 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 [17]. Values of NMSE lower than 1 indicate that the MLP provided better predictions than the mean value at least.

The above methodology was applied to each learning algorithm (BP and LM). The performance information stored in the meta-example is the difference between the NMSE values obtained by the trained MLP (with optimal number of nodes) on the test subset, respectively using the BP and the LM algorithm.

3.2 Meta-Learner

The set of meta-examples is used to build a meta-regressor which will predict the performance measure Y for new problems. In our prototype, we deployed the Linear Regression (LR) model, which has some advantages to Meta-Learning. First, it produces in a short time a model which is easy to interpret [3]. Second, as it will be seen, an Active Learning method can be straightforwardly derived from the LR model using the confidence intervals estimated for its predictions.

The LR model is described in different textbooks (e.g., [18]). Here, we provide a short description of the LR model, which is adequate for understanding our work. Given an input problem e described by the vector $x = (x^1, \ldots, x^p)$, the prediction $\widehat{y} = \widehat{Y}(e)$ provided by the meta-regressor can be defined as:

$$\widehat{y} = \widehat{\beta}_0 + \sum_{j=1}^{p} \widehat{\beta}_j * x^j \tag{2}$$

The model may be more conveniently written in a matrix notation as:

$$\widehat{y} = \mathbf{x}\widehat{\beta} \tag{3}$$

where

$$\mathbf{x} = \left[1 \; x^1 \; x^2 \ldots x^p \right]$$

and

$$\widehat{\beta} = \left[\widehat{\beta}_0 \; \widehat{\beta}_1 \ldots \widehat{\beta}_p \right]^T .$$

The parameters represented in $\widehat{\beta}$ are derived by the least-squares method which minimizes the sum of squared error for the training data. The least square estimator $\widehat{\beta}$ is computed from the n training meta-examples in ME as:

$$\widehat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{y} \tag{4}$$

where

$$\mathbf{X} = \begin{bmatrix} 1 \; x_1^1 \; x_1^2 \; \ldots \; x_1^p \\ 1 \; x_2^1 \; x_2^2 \; \ldots \; x_2^p \\ \vdots \;\; \vdots \;\;\; \vdots \;\;\;\;\;\;\;\; \vdots \\ 1 \; x_n^1 \; x_n^2 \; \ldots \; x_n^p \end{bmatrix}$$

and

$$\mathbf{y} = \begin{bmatrix} y_1 \; y_2 \; \ldots \; y_n \end{bmatrix}^T .$$

3.3 Active Learning

As seen, the ML module generates a meta-regression model from a set of *labeled* meta-examples, associated to the set of problems E in which the performance information is known. The AL module receives a set of *unlabeled* problems $\widetilde{E}$ in which the algorithms were not yet evaluated. The AL module incrementally selects unlabeled problems in $\widetilde{E}$ to be used for generating new meta-examples.

In the prototype, the AL module deployed an uncertainty-based active learning method. In this approach, the Meta-Regressor initially generates its predictions for each unlabeled example $\widetilde{e} \in \widetilde{E}$ by using the estimated model (see section 3.2). A degree of prediction uncertainty $\mathcal{S}(\widetilde{e}|E)$ is assigned to each unlabeled example. Finally, the unlabeled example with the highest prediction uncertainty is selected. The uncertainty-based approach can be directly applied to the LR model, since confidence intervals for its responses can be reliably estimated (see [18]). A $100(1-\alpha)$ percent confidence interval of $\widehat{y}$ is defined as:

$$\widehat{y} - t_{\alpha/2,n-p-1}\sqrt{V(\widehat{y})} \leq y \leq \widehat{y} + t_{\alpha/2,n-p-1}\sqrt{V(\widehat{y})} \tag{5}$$

In the above equation, $V(\widehat{y})$ is the variance of the prediction $\widehat{y}$ which is estimated as:

$$V(\widehat{y}) = \widehat{\sigma}^2 \mathbf{x}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x} \tag{6}$$

where $\widehat{\sigma}^2$ is the residual mean square of the LR model, defined as:

$$\widehat{\sigma}^2 = \frac{\mathbf{y}^T\mathbf{y} - \widehat{\beta}^T\mathbf{X}^T\mathbf{y}}{n-p-1} \tag{7}$$

The size of the confidence interval defined in Eq. (5), which is $2 * t_{\alpha/2,n-p}\sqrt{V(\widehat{y})}$, reflects the uncertainty of the prediction provided by the LR model. Since this value for different unlabeled examples solely depends on $V(\widehat{y})$, we defined the degree of prediction uncertainty in the AL module as:

$$\mathcal{S}(\widetilde{e}|E) = V(\widehat{y}) \tag{8}$$

where $\widehat{y}$ is the prediction provided by the meta-regressor for the problem $\widetilde{e}$. The AL module then selects, for generating a new meta-example, the problem $\widetilde{e}^* \in \widetilde{E}$ with highest uncertainty. A new meta-example is then generated from $\widetilde{e}^*$ as described in section 3.1.

4 Experiments and Results

In our work, we performed experiments with a set of meta-examples generated from 50 regression problems, available in the WEKA project[2]. On average, the collected datasets presented 4,392 examples and 13.92 attributes. We observed in these datasets a large variability in both the number of examples and attributes, which is convenient to Meta-Learning studies.

A leave-one-out experiment was performed to evaluate the performance of the meta-regressor, also varying the number of meta-examples provided by the AL module. 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 used to predict the algorithm performance on the test problem left out, and the obtained squared error is registered. Hence, a curve with 49 squared errors is produced for each test problem. Finally, the curve of error rates obtained by ML is summarized by computing the NMSE measure over the 50 steps of the leave-one-out experiment.

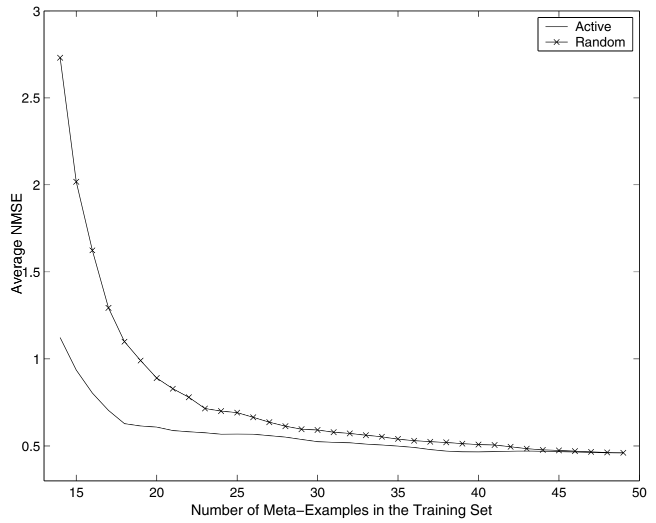

Fig. 2. Average curves of error rates for active and random methods. For simplicity, high rates ($NMSE > 3$) obtained in the early part of the curves were not displayed.

As a basis of comparison, the same above experiment was applied by using a random method for selecting unlabeled problems. Despite its simplicity, the random method has the advantage of performing a uniform exploration of the

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

example space, and hence, it has been very competitive when compared to other active methods [19]. Finally, we highlight that the leave-one-out experiment was performed in 30 different runs for both the AL method using uncertainty sampling and the random procedure. The performance of the meta-regressor was evaluated by averaging the curves of error rates over the 30 runs.

Figure 2 presents the average curve of error rates for the active and the random methods. 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 active method were, in general, lower than the error rates obtained by deploying the random method. From 17 meta-examples included in the training set, the NMSE values obtained by using the active method were lower than 1 (a threshold which indicates the viability of using the meta-regressor). By considering the random method, NMSE values lower than 1 were achieved only from 24 included meta-examples. In the experiments, we also compared the active method to the random method in statistical terms, by applying a t-test (95% of confidence) to the difference of error rates obtained by the meta-regressor in each point of the curves of error rates. In this evaluation, the active method obtained a statistical gain over the random method from 15 to 34 meta-examples included in the training set, which represent 40.82% of the 49 points considered in the curves.

5 Conclusion

In this paper, we presented the use of Active Learning to support the generation of training examples for Meta-Regression. In our proposal, Active Learning methods are used to reduce the cost of generating a good set of meta-examples while maintaining the performance of the meta-regressor. We highlight that our work brings contributions in both fields the Meta-Learning and the Active Learning.

In order to verify the viability of our proposal, we implemented a prototype to select meta-examples for a LR meta-regressor. The prototype was evaluated in a task of predicting the performance of MLPs, achieving satisfactory experimental results. Despite the advantages of LR as meta-learner, a large number of regression algorithms could be deployed. For instance, we intend to evaluate the use of Support Vector Machines (SVMs) as meta-regressors. Consequently, we will adapt active techniques previously proposed for standard SVMs (e.g., [20]). We also intend to propose active methods for other Meta-Learning approaches which include, for instance, ranking approaches (e.g., [8,9]).

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

References

1. Giraud-Carrier, C., Vilalta, R., Brazdil, P.: Introduction to the special issue on meta-learning. Machine Learning 54(3), 187–193 (2004)
2. Vilalta, R., Giraud-Carrier, C., Brazdil, P., Soares, C.: Using meta-learning to support data- mining. Intern. Journal of Computer Science Application I(31), 31–45 (2004)

3. Gama, J., Brazdil, P.: Characterization of classification algorithms. In: Pinto-Ferreira, C., Mamede, N.J. (eds.) EPIA 1995. LNCS, vol. 990, pp. 189–200. Springer, Heidelberg (1995)
4. Bensusan, H., Alexandros, K.: Estimating the predictive accuracy of a classifier. In: Proc. of the 12th European Conference on Machine Learning, pp. 25–36 (2001)
5. Koepf, C.: Meta-regression: performance prediction. In: Meta-Learning: Strategies, Implementations, and Evaluations for Algorithm Selection, pp. 89–106 (2006)
6. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. Machine Learning 15, 201–221 (1994)
7. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323, 533–536 (1986)
8. Brazdil, P., Soares, C., da Costa, J.: Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. Machine Learning 50(3), 251–277 (2003)
9. Kalousis, A., Gama, J., Hilario, M.: On data and algorithms - understanding inductive performance. Machine Learning 54(3), 275–312 (2004)
10. Aha, D.: Generalizing from case studies: A case study. In: Proceedings of the 9th International Workshop on Machine Learning, pp. 1–10. Morgan Kaufmann, San Francisco (1992)
11. Kalousis, A., Hilario, M.: Representational issues in meta-learning. In: Proceedings of the 20th International Conferente on Machine Learning, pp. 313–320 (2003)
12. Prudêncio, R.B.C., Ludermir, T.B.: Meta-learning approaches to selecting time series models. Neurocomputing 61, 121–137 (2004)
13. Leite, R., Brazdil, P.: Predicting relative performance of classifiers from samples. In: Proceedings of the 22nd International Conference on Machine Learning (2005)
14. Prudêncio, R.B.C., Ludermir, T.B.: Selective generation of training examples in active meta-learning. Intern. Journal of Hybrid Intelligent Systems 5, 59–70 (2008)
15. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Quarterly Journal of Applied Mathmatics II(2), 164–168 (1944)
16. Prechelt, L.: A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Universitä Karlsruhe, Germany (1994)
17. Soares, C., Brazdil, P., Kuba, P.: A meta-learning approach to select the kernel width in support vector regression. Machine Learning 54(3), 195–209 (2004)
18. Montgomery, D., Peck, E., Vining, G.: Introduction to Linear Regression Analysis, 4th edn. Wiley, Chichester (2006)
19. Lindenbaum, M., Markovitch, S., Rusakov, D.: Selective sampling for nearest neighbor classifiers. Machine Learning 54, 125–152 (2004)
20. Schohn, G., Cohn, D.: Less is more: Active learning with support vector machines. In: Proc. of the 17th Intern. Conference on Machine Learning, pp. 839–846 (2000)