Using Support Vector Machines to Predict the Performance of MLP Neural Networks

Conference Paper · October 2008

DOI: 10.1109/SBRN.2008.30 · Source: DBLP

3 authors, including:

Ricardo B. C. Prudêncio
Federal University of Pernambuco
90 PUBLICATIONS **1,699** CITATIONS

SEE PROFILE

Teresa Bernarda Ludermir
Federal University of Pernambuco
436 PUBLICATIONS **4,788** CITATIONS

SEE PROFILE

Using Support Vector Machines to Predict the Performance of MLP Neural Networks

Ricardo B. C. Prudêncio, Silvio B. Guerra, Teresa B. Ludermir
Center of Informatics, Federal University of Pernambuco
Pobox 7851 - CEP 50732-970 - Recife (PE) - Brazil
rbcp@cin.ufpe.br, silvio.guerra@gmail.com
tbl@cin.ufpe.br

Abstract

In this work, we investigated the use of Support Vector Machines (SVM) to predict the performance of learning algorithms based on features of the learning problems, in a kind of Meta-Learning. Experiments were performed in a case study in which SVM regressors with different kernel functions were used to predict the performance of Multi-Layer Perceptron (MLP) networks. The results obtained on a set of 50 learning problems revealed that the SVMs obtained better results in predicting the MLP performance, when compared to benchmark algorithms applied in previous work.

1 Introduction

The adequate choice of learning algorithms is an important aspect to the success of the Machine Learning applications [1]. Algorithm selection can be performed by empirically evaluating the candidate algorithms using the avaliable data, which can be a costly solution [2]. Algorithm selection can also be guided by expert rules, however such expert knowledge is not always easy to acquire [3].

In order to avoid the limitations of the above solutions, different authors have investigated the use of Meta-Learning to automatically acquiring knowledge for predicting learning performance, thus supporting algorithm selection [1, 2, 3, 4, 5, 6, 7, 8, 9]. The knowledge in Meta-Learning is acquired from a set of meta-examples which store features of learning problems and the information about the performance obtained by the candidate algorithms (the base-learners) on the problems. The knowledge in this case can be represented as a learning model (i.e., the meta-learner) that relates features of the problems and the performance of the learning algorithms.

Meta-Regression [3] is a specific Meta-Learning approach which uses a regression algorithm to predict the value of a chosen performance measure (e.g., classification error) of the candidate algorithms based on the features of the problems. The meta-learner is a regression model that may be used to select the best candidate algorithm considering the highest predicted performance measure. Different regression algorithms have been applied in this context, such as decision trees, linear regression and instance-based algorithms [3, 5, 6].

In the current work, we investigated the use of Support Vector Machines (SVMs) [10] to Meta-Regression. In literature, SVMs have been successfully applied to many different problems, achieving very competitive results when compared to other machine learning algorithms [11]. In the context of Meta-Regression, there is no previous work that evaluated the use of SVMs to predict the performance of learning algorithms. Experiments were performed in a case study which consists of predicting the performance of Multi-Layer Perceptron (MLP) networks [12]. A set of 50 meta-examples was produced from the application of the MLP on 50 different learning problems.

In the meta-level, we applied SVMs with different kernel functions (among polynomial and RBF kernels) to predict the value of the Normalized Mean of Squared Errors (NMSE) of the MLPs. We also applied the Jaakkola heuristic [13] to automatically define the parameter of the RBF kernel. The predictions of the NMSE were based on a set of 10 pre-defined features of the learning problems (e.g., number of training examples and correlation between attributes). The performance of the SVMs was evaluated in a leave-one-out experiment performed on the 50 meta-examples.

As a basis of comparison, we also performed experiments using three different benchmark regression algorithms used in previous work to Meta-Regression: the M5 algorithm (decision trees), the linear regression, and the 1-nearest neighbor algorithm. The performed experiments revealed that the SVMs with simple polynomial kernels ob-

tained better performance in the meta-regression task when compared to the benchmark algorithms. The experiments also revealed that good performance can also be achieved by using SVM with RBF kernels, depending, in this case, on an adequate choice of the kernel's parameters.

Section 2 brings an introduction to Meta-Learning, including the Meta-Regression approach. Section 3 presents details of the proposed work, as well as the case study. Section 4 brings the experiments and results. Finally, section 5 presents some final conclusions.

2 Meta-Learning

Meta-Learning is the automatic process of acquiring knowledge relating the performance of learning algorithms to the features of the learning problems [1]. Each *meta-example* is related to a learning problem and stores: (1) the features describing the problem, called *meta-features*; and (2) the performance information of one or more algorithms when applied to the problem. The *meta-learner* is a learning system that receives 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 [7, 8]. 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 [4]. 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 [2, 4, 9, 14, 15]), 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, 6] tries to directly predict the numerical value of accuracy (or alternatively the error rate) of each candidate algorithm. In this case, the meta-examples store as performance information the numerical values of accuracy 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 accuracy or to provide a ranking of algorithms based on the order of predicted accuracies.

In [3], the authors evaluated different algorithms as meta-regressors, including linear regression models, piecewise linear models, decision trees and instance-based regression. In [6], 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 Meta-Regression with SVMs

In the current work, we investigate the use of Support Vector Machines (SVMs) in the context of Meta-Regression. SVM is a state-of-the-art algorithm in the Machine Learning field, successfully applied to different classification and regression problems [10, 11]. Previous work in Meta-Regression has applied different regression methods to produce meta-learners (as seen in section 2), yielding relative success. However, to the best of our knowledge, there is no evaluation of the use of SVMs in this task.

Figure 1 brings the architecture of the Meta-Regression which summarizes our proposal. Each meta-example is composed by the meta-features of a learning task and the performance information derived from the empirical evaluation of the learning algorithm on the task. The set of generated meta-examples is given as input to the SVM algorithm, which will produce a regression model responsible for predicting the algorithm's performance for new problems based on its meta-features.

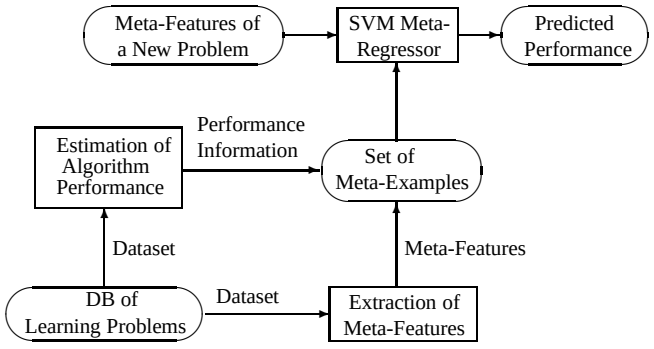

Figure 1. Architecture of Meta-Regression

A case study was performed in our work in a meta-learning task which corresponds to predict the performance of Multi-Layer Perceptron (MLP) networks [12] in regression problems. In this case study, we generated a set of 50 meta-examples from the application of the MLP in 50 regression problems (see section 3.1). Each meta-example is associated to a single problem and stores: (1) the value of 10

descriptive meta-features (see section 3.2); and (2) the test error obtained by the MLP when evaluated in the regression problem (see section 3.3). The set of meta-examples is given as input to a regression algorithm which will be able to predict the error of the MLP for new problems only based on the meta-attributes of the problems.

In the following sections, we provide details about the construction of the set of meta-examples, as well as the details on the SVMs used for Meta-Regression.

3.1 Datasets

In order to generate meta-examples, we collected 50 datasets corresponding to 50 different regression problems, available in the WEKA project [1]. 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 training examples and the number of attributes. This variation may be convenient to Meta-Learning studies, since it is expected that the algorithms present significantly different patterns of performance, depending on the problem being solved.

The attributes in the original datasets were normalized to the [-1; +1] interval, aiming a better treatment by the MLP.

3.2 Meta-Features

A total number of 10 meta-features was used to describe the datasets of regression problems:

1. LogE - Log of the number of training examples;
2. LogEA - Log of the ration between number of training examples and number of attributes;
3. MinCT, MaxCT, MeanCT and StdCT - Minimum, maximum, mean and standard deviation of the absolute values of correlation between predictor attributes and the target attribute;
4. MinCAtr, MaxCAtr, MedCAtr and DesvCAtr - Minimum, maximum, mean and standard deviation of the absolute values of correlation between pairs of predictor attributes.

The meta-feature LogE is an indicator of the amount of data available for training, and LogEA, in turn, indicates the dimensionality of the dataset. The meta-features MinCT, MaxCT, MedCT and DesvCT indicate the amount of relevant information available to predict the target attribute. The meta-features MinCAtr, MaxCAtr, MedCAtr and DesvCAtr, in turn, indicate the amount of redundant information in the dataset.

[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/

3.3 Performance Information

The final step to generate the meta-examples is to evaluate the performance of the MLP in the 50 collected regression problems. From this evaluation, we can produce the performance information stored in the meta-examples which will correspond to the target attribute in the Meta-Regression task. In order to measure the performance of the MLP in each problem, the following methodology of evaluation was applied.

Each dataset was divided in the training, validation and test subsets, in the proportion of 50%, 25% and 25%, respectively. In our work, we applied the standard Backpropagation (BP) algorithm [12] to train the MLP network[2]. The optimal number of hidden nodes[3] was empirically defined, by testing the values 1, 2, 4, 8, 16 and 32. For each number of hidden nodes, the MLP is trained 10 times with random initial weights. The stopping criteria of the training process followed the benchmarking rules provided in [17].

The optimal number of hidden nodes was finally chosen as the value in which the trained MLP obtained the lowest average NMSE (Normalized Mean of Squared Errors) on the validation subset over the 10 training runs. The NMSE measure is defined as:

$$NMSE = \frac{\sum_{i=1}^{n}(y_i - \widetilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \quad (1)$$

In the equation, n is the number of examples in the validation subset, y_i e $\widetilde{y}_i$ are respectively the target and the predicted value for example i, and $\overline{y}$ is the average of the target attribute.

The performance information which is stored in the meta-example is the average NMSE obtained by the trained MLP (with optimal number of hidden nodes) on the test subset. According to [18], a property of NMSE which is interesting for Meta-Learning is that the values have no scale and are comparable across different datasets (which it would not be feasible if we had used a non-normalized error measure). Values of NMSE lower to 1 indicate that the MLP provided better predictions than the mean value. Values higher than 1 indicate that the MLP was not useful in the regression problem.

Figure 2 presents a box plot of the NMSE test values obtained by BP on the 50 problems. The median of the NMSE values was 0.75, which indicates that the performance of the BP was, in general, adequate considering the 50 problems. The size of the box plot indicates, however, that the BP algorithm may present a very different behavior depending on the considered problem.

[2]The BP algorithm was implemented by using the NNET Matlab toolbox [16]. Learning rates were defined by default.

[3]The MLP was defined with only one hidden layer.

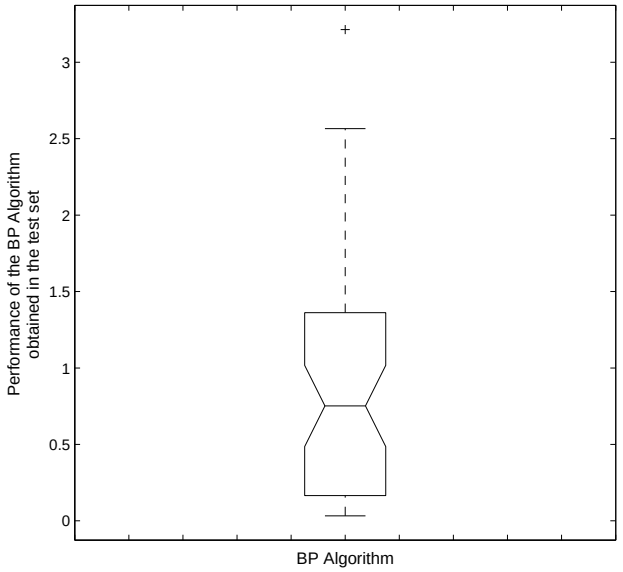

Figure 2. Box Plot of the NMSE test values obtained by the BP algorithm in the 50 problems

We highlight here that there are different algorithms to train MLPs that could have been used to improve performance. However, our aim in this work is not to achieve the best possible performance with MLPs but to *predict* the learning performance. Other learning algorithms to train MLPs (such as the Levenberg-Marquardt algorithm [19]) can be applied in the future as new case studies, possibly with more effective strategies to train the networks.

3.4 Meta-Regressor

In our case study, the Meta-Regression task is to predict the NMSE measure of the MLP based on the features of the problems given as input. In our experiments, this task is dealt with SVM regressors.

An important aspect of the SVMs is the chosen kernel function. In our work, we evaluated the use of two different types of kernel functions in the SVMs. First, we evaluated homogeneous polynomial kernels represented as:

$$K(x, x') = (x \cdot x')^p \tag{2}$$

In our experiments, we used the values $p = 1$ (linear kernel) and $p = 2$ (quadratic kernel), which are the more common options for polynomial kernels. We also deployed in our work a Radial Basis Function (RBF) kernel in the form:

$$K(x, x') = e^{-\gamma \cdot \|x - x'\|^2} \tag{3}$$

The RBF kernel is a non-linear function that, in comparison to the polynomial kernels, it is expected to handle more complex relationships between predictor and target attributes. In our experiments, we fixed the values of γ in the range [0.01; 10.24]. We evaluated 11 different values in this interval following a geometric progression with factor 2. Hence, the set of evaluated values is the sequence 0.01, 0.02, 0.04, ..., 5.12, 10.24.

Besides the above sequence of evaluated values, we also performed experiments in our work by deploying the Jaakkola heuristic [13] for defining the value of γ. The parameter γ is defined in this heuristic based on the distances between examples in the attribute space. Initially, for each example $\mathbf{x} = (x^1, \ldots, x^j)$, we computed the minimum distance of $\mathbf{x}$ to the other examples:

$$d_{\mathbf{x}} = min_i \sqrt{\sum_j (x^j - x_i^j)^2} \tag{4}$$

where x_i^j is the value of the j-th attribute in the example $\mathbf{x}_i$. The parameter γ is then defined as:

$$\gamma = \frac{1}{2\overline{d_{\mathbf{x}}}^2} \tag{5}$$

where $\overline{d_{\mathbf{x}}}$ is the average value of the minimum distances over all training examples in the attribute space. The value of γ obtained by using the Jaakkola heuristic was 0.1392, considering our set of meta-examples.

Obviously, there are other possibilities of kernel functions (e.g., sigmoid kernels), as well as parameter settings that we intend to experiment in future work. Besides, automated procedures can be used in the future to define the values of these parameters [20].

In this work, we deployed the implementation of SVMs provided in the WEKA environment [21], which applies the sequential minimal optimization algorithm proposed by [10] for training the SVM regression model.

4 Experiments and Results

In this section, we evaluated the performance of the Meta-Regression process for the generated set of meta-examples. In our experiments, the SVM meta-regressor was evaluated by using the leave-one-out procedure. Two different criteria were used to evaluated the meta-learner: (1) the Normalized Mean of Squared Errors (NMSE) computed for the predictions of the meta-learner in the test meta-examples; and (2) the Correlation (COR) between the predictions generated by the meta-learner and the true values of the target attribute stored in the test meta-examples.

As a basis of comparison, we performed experiments using three different methods as meta-regressors:

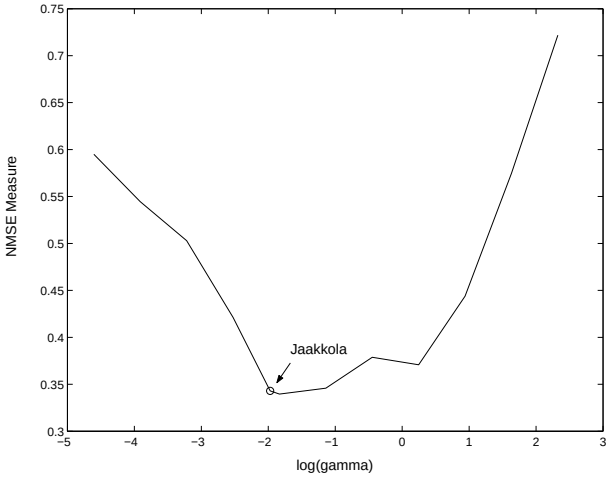

Figure 3. NMSE test values obtained by the meta-learner for different values of the kernel parameter (γ).

1. M5 algorithm: which was proposed by Quinlan [22] to induce regression trees;

2. 1-Nearest Neighbor (1-NN) algorithm: a special case of instance-based learning algorithm [23];

3. Linear Regression (LR) model.

All these methods were already used in previous work as meta-regressors (see, for instance, [3]). We highlight that these algorithms are representatives of different families of regression methods, providing different inductive biases for the learning problem being solved in the case study.

As it can be seen in Table 1, the SVM regressor with polynomial kernels (linear and quadratic) obtained better results when compared to the benchmark methods M5, 1-NN and LR. The best result among the polynomial kernels was obtained by the quadratic kernel, which also yielded a competitive result when we include the SVM with RBF kernel in the comparison.

Considering the RBF kernel, we observed that the performance of the SVM was very sensitive to the value of parameter γ (see Figure 3). For $\gamma = 0.16$, the SVM obtained the best result over all evaluated algorithms and parameter settings. For some values of γ, however, the performance of the SVM with RBF kernel was worse than the performance obtained by the benchmark methods. This result indicates that an adequate choice of the parameter of RBF kernels (i.e., the model selection) is an aspect that should be more carefully addressed in the case of SVM meta-regressors. In fact, when we used the Jaakkola heuristic to model selection, the SVM obtained a performance which is similar to the best result obtained with $\gamma = 0.16$ (see Figure 3).

Table 1. Results obtained by Meta-Regression in the leave-one-out experiment.

	NMSE	COR
SVM (linear)	0.486	0.731
SVM (quadratic)	0.457	0.758
SVM (RBF, $\gamma = 0.01$)	0.595	0.701
SVM (RBF, $\gamma = 0.02$)	0.545	0.716
SVM (RBF, $\gamma = 0.04$)	0.503	0.718
SVM (RBF, $\gamma = 0.08$)	0.421	0.763
SVM (RBF, $\gamma = \mathbf{0.16}$)	**0.340**	**0.817**
SVM (RBF, $\gamma = 0.32$)	0.346	0.812
SVM (RBF, $\gamma = 0.64$)	0.379	0.792
SVM (RBF, $\gamma = 1.28$)	0.371	0.794
SVM (RBF, $\gamma = 2.56$)	0.444	0.746
SVM (RBF, $\gamma = 5.12$)	0.575	0.653
SVM (RBF, $\gamma = 10.24$)	0.722	0.528
SVM (RBF, Jaakkola)	**0.343**	**0.815**
M5	0.605	0.631
1-NN	0.539	0.710
LR	0.595	0.658

Model selection is in fact a relevant topic of research in SVMs, which can be handled, for instance, by considering the characteristics of the data [10, 18, 24] and by deploying search techniques [20, 25, 26]. Such strategies will be considered in future work to model selection in our case study.

5 Conclusion

In this work, we investigated the use of SVMs to Meta-Regression, aiming to predict the performance of learning algorithms based on the features of the learning problems. Experiments were performed in a case study to predict the performance of MLPs applied to regression problems. The results of experiments on a set of 50 meta-examples revealed that the performance of the SVM meta-regressor was in general better than the performance obtained by the benchmark algorithms applied as a basis of comparison.

Although we applied, in the case study, polynomial and RBF kernels which are widely used in the literature, different other functions can be evaluated. Furthermore, the kernel parameters could be optimized by using more sophisticated approaches. We also include as future work, the evaluation of SVM meta-regressors in new case studies related to other machine learning algorithms.

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

References

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

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

[3] Gama, J. and Brazdil, P. (1995). Characterization of classification algorithms. In *Proceedings of the 7th Portuguese Conference on Artificial Intelligence*, pages 189–200.

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

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

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

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

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

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

[10] Smola, A. and Scholkopf, B. (2004). A tutorial on support vector regression. In *Statistics and Computing*, 14(3):199-222.

[11] Wang, L. (2005). *Support Vector Machines: Theory and Applications*. Springer, Berlin.

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

[13] Jaakkola, T., Diekhans, M., and Haussler, D. (1999). Using the Fisher kernel method to detect remote protein homologies. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 149-158.

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

[15] Leite, R. and Brazdil, P. (2005). Predicting relative performance of classifiers from samples. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 497–503.

[16] Demuth, H. and Beale, M. (1993). *Neural Network Toolbox: For use with MATLAB: User's Guide*. The Mathworks.

[17] Prechelt, L. (1994). A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Fakultät für Information, Universitä Karlsruhe, Karlsruhe, Germany.

[18] Soares, C., and Brazdil, P. (2004). A Meta-learning method to select the kernel width in support vector regression. *Machine Learning*, 54:195–209.

[19] Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathematics*, II(2):164–168.

[20] Chapelle, O., Vapnik, V., Bousquet, O. and Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. In *Machine Learning*, 46(1):131-159.

[21] Witten, I. H. and Frank, E., editors (2003). *WEKA: machine learning algorithms in Java*. University of Waikato, New Zealand.

[22] Quinlan J. R., (1992). Learning with continuous classes. In *Proceedings of the Australian Joint Conference on Artificial Intelligence*, pages 343–348.

[23] Aha, D. and Kibler, D. (1991). Instance-based learning algorithms. *Machine Learning*, 6(3):37–66.

[24] Kuba, P., Brazdil, P., Soares, C., and Woznica, A. (2002). Exploiting sampling and meta-learning for parameter setting support vector machines. In *Proceedings of the IBERAMIA 2002*, pages 217-225.

[25] Cawley, G. (2001). Model Selection for Support Vector Machines via Adaptive Step-Size Tabu Search. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 434–437.

[26] Lessmann, S., Stahlbock, R. and Crone, S. (2006). Genetic Algorithms for Support Vector Machine Model Selection. In *International Joint Conference on Neural Networks*, pages 3063–3069.