Agnostic Learning with Ensembles of Classifiers

Jörg D. Wichard

***Abstract*— We present a method for building ensembles of models in order to build proper classifiers. The main advantage of our method is an automated model selection procedure and an automated model parameter estimation. The method is an extension of the classical bagging and the K-fold-cross-validation approach.**

I. Introduction

Ensemble building is a common way to improve the performance of the resulting model for classification and regression tasks. An ensemble of individual models performs in general better than a single model in the average. This was first introduced in the neural networks community since it was discovered, that a combination of several Neural Networks can reduce the variance of the average regression model [1], [2], [3]. The extension to classification problems was straightforward after the formulation of a bias-variance decomposition for zero-one loss functions [4], [5]. The key feature of the ensemble approach is the introduction of model diversity [6], [7], [8] that helps to reduce the variance of the resulting ensemble model. There are several ways to achieve diverse models like the well known bootstrap aggregating or 'bagging' (see Breiman [9]) where the models are trained on different subsets of the training data or heterogeneous ensembles, that consist of several different model classes like Neural Networks, nearest-neighbor models, decision trees, etc [10], [11].
Sometimes it might be convenient to focus on a specific model class, in particular if one has to deal with large data sets and the time for model training or processing is critical (linear models and trees). If we consider a supervised learning problem with n training examples of the form $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ from an unknown function $y = f(\mathbf{x})$. The $\mathbf{x}$ values are usually d-dimensional vectors that are called *'input-features'* while the y values are continuous in the case of regression and discrete *'class labels'* in the case of classification. If $y \in \{0, 1\}$ we call it a *'binary classification problem'*. A *'classifier'* is a hypothesis about the unknown function $y = f(\mathbf{x})$ in the sense, that given some new values $\mathbf{x}^*$ it predicts the corresponding class labels y^*. A classifier ensemble is a set of single classifiers whose individual predictions are combined in order to classify new data examples.

Jörg D. Wichard is with the Institute of Molecular Pharmacology Molecular Modelling Group, Robert Rössle Straße 10, D-13125 Berlin-Buch, and with the Charité Institute of Medical Informatics, Hindenburgdamm 30, 12200 Berlin, Germany (email: JoergWichard@web.de).

II. Ensembles

The average output of several different models $f_i(\mathbf{x})$ marks the ensemble model

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{K} \omega_i f_i(\mathbf{x}), \tag{1}$$

where we assume that the model weights ω_i sum to one $\sum_{i=1}^{K} \omega_i = 1$. There are several suggestions concerning the choice of the model weights (see Perrone et al. [3] or Hashem et al. [12]). We decided to use uniform weights with $\omega_i = 1/K$ for the sake of simplicity and not to run into over-fitting problems as reported by Krogh et al. [8].
The central feature of the ensemble approach is the generalization ability of the resulting model. It is related to finding the right balance between model complexity and generalization in order to avoid overfitting as depicted in Figure 1.

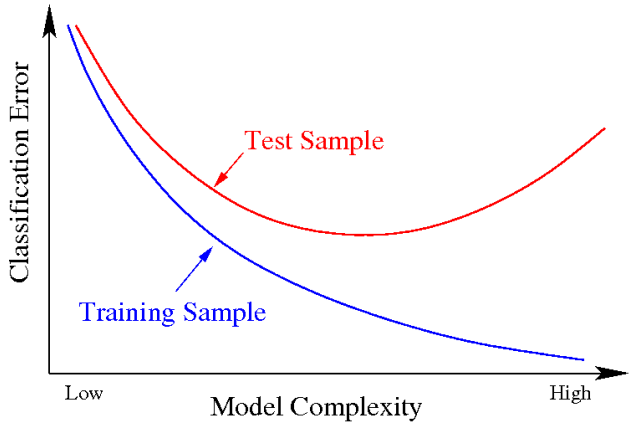

Fig. 1. The balance between model complexity and generalization ability of the final classification model.

In the case of regression models (with continuous output values) it was shown, that the generalization error of the ensemble is in the average case lower than the mean of the generalization error of the single ensemble members (see Krogh 1995 [6]). This holds in general, independent of the model class, as long as the models constituting the ensemble are diverse with respect to the hypothesis of the unknown function. In the case of (binary) classification models the situation was not so clear because the classical bias-variance decomposition of the squared error loss in regression problems (Geman et al. [2]) had to be extended to the zero-one loss function. There are several approaches dealing with this problem, see Kong and Dietterich [13], Kohavi [4] or Domingos [5].
The zero-one loss function is not the only possible choice for

classification problems. If we are interested in a likelihood whether a sample belongs to one class or not, we can use the error loss from regression and consider the binary classification problem as a regression problem that works on two possible outcomes. In practice, many classifiers are trained in that way.

Our ensemble approach is based on the observation that the generalization error of an ensemble model could be improved if the models on which averaging is done disagree and if their fluctuations are uncorrelated [8]. To see this we have to investigate the contribution of the single model in the ensemble to the generalization error. We consider the case where we have a given data set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ and we want to find a function $f(\mathbf{x})$ that approximates y at new observations of $\mathbf{x}$. These observations are assumed to come from the same source that generated the training set D, i.e. from the same (unknown) probability distribution P. It should be noted that f depends also on D. The expected generalization error $Err(\mathbf{x}, D)$ given a particular $\mathbf{x}$ and a training set D is

$$Err(\mathbf{x}, D) = E[(y - f(\mathbf{x}))^2 | \mathbf{x}, D], \tag{2}$$

where the expectation $E[\cdot]$ is taken with respect to the probability distribution P. We are now interested in

$$Err(\mathbf{x}) = E_D[Err(\mathbf{x}, D)], \tag{3}$$

where the expectation $E_D[\cdot]$ is taken with respect to all possible realizations of training sets D with fixed sample size N. According to [2] the bias/variance decomposition of $Err(\mathbf{x})$ is

$$\begin{aligned} Err(\mathbf{x}) &= \sigma^2 + (E_D[f(\mathbf{x})] - E[y|\mathbf{x}])^2 \\ &\quad + E_D[(f(\mathbf{x}) - E_D[f(\mathbf{x})])^2] \\ &= \sigma^2 + (Bias(f(x)))^2 + Var(f(x)), \end{aligned} \tag{4}$$

where $E[y|\mathbf{x}]$ is the deterministic part of the data and σ^2 is the variance of y given $\mathbf{x}$. Balancing between the bias and the variance term is a crucial problem in model building. If we try to decrease the bias term on a specific training set, we usually increase the variance term and vice versa [14]. We now consider the case of an ensemble average $\hat{f}(\mathbf{x})$ consisting of K individual models as defined in Equ. (1). If we put this into Equ. (4) we get

$$Err(\mathbf{x}) = \sigma^2 + Bias(\hat{f}(x))^2 + Var(\hat{f}(x)), \tag{5}$$

and we can have a look at the effects concerning bias and variance. The bias term in Equ. (5) is just the average of the biases of the individual models in the ensemble. So we should not expect a reduction in the bias term compared to single models. The variance term of the ensemble could be decomposed in the following way:

$$\begin{aligned} Var(\hat{f}) &= E\left[(\hat{f} - E[\hat{f}])^2\right] \\ &= E[(\sum_{i=1}^{K} \omega_i f_i)^2] - (E[\sum_{i=1}^{K} \omega_i f_i])^2 \\ &= \sum_{i=1}^{K} \omega_i^2 \left(E\left[f_i^2\right] - E^2\left[f_i\right]\right) \\ &\quad + 2 \sum_{i<j} \omega_i \omega_j \left(E\left[f_i f_j\right] - E\left[f_i\right] E\left[f_j\right]\right), \end{aligned} \tag{6}$$

where the expectation is taken with respect to D. The first sum in Equ. (6) marks the lower bound of the ensemble variance and is the weighted mean of the variances of the ensemble members. The second sum contains the cross terms of the ensemble members and disappears if the models are completely uncorrelated [8]. So the reduction in the variance of the ensemble is related to the degree of independence of the single models [7]. This is the key feature of the ensemble approach.

III. Cross-Validation and Model Selection

Our model selection scheme is a mixture of bagging [9] and cross-validation. *Bagging* or *Bootstrap aggregating* was proposed by Breiman [9] in order to improve the classification by combining classifiers trained on randomly generated subsets of the entire training sets. We extended this approach by applying a cross-validation scheme for model selection on each subset.

In K-fold cross-validation, the data set is partitioned into K subsets. Of these K subsets, a single subset is retained as the validation data for testing the model, and the remaining $K1$ subsets are used for model training. The cross-validation process is then repeated K times with each of the K subsets used only once as the validation data. The K results from the folds then can be averaged to produce a single estimation.

If we lack relevant problem-specific knowledge, cross-validation methods could be used to select a classification method empirically [15]. This is a common approach because it seems to be obvious that no classification method is uniformly superior, see for example Quinlan [16] for a detailed study. It is also a common approach to select the model parameters with cross-validation [17]. The idea to combine the models from the K cross-validation folds (stacking) was described by Wolpert [18]. We suggest to train different model classes on each CV-fold, to select the best performing model on the validation set and to combine the models from the K-folds. We call this a heterogeneous ensemble and applied this method effectively to regression problems [19], classification tasks [10] and in several modelling competitions [20], [21].

Our model selection scheme works as follows: For the K-fold CV the data is divided K-times into a *'training set'* and a *'test set'*, both sets containing randomly drawn subsets of

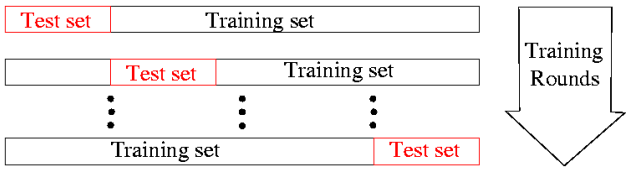

Fig. 2. For every partition of the cross-validation, the data is divided in a training and a test set.

the data without replications. The ratio

$$R = \frac{\text{\# training samples}}{\text{\# test samples}} \tag{7}$$

ranges from $\frac{70}{30}$ up to $\frac{50}{50}$. In every CV-fold we train several different models classes with a variety of model parameters (see Section V for an overview of the models and the related model parameters). In each fold we select only one model to become a member of the final ensemble (namely the best model with respect to the test set). This means, that all models have to compete with each other in a fair tournament because they are trained and validated on the same data set. The models with the lowest classification error in each CV-fold are taken out and added to the final ensemble, receiving the weight $\omega_i = \frac{1}{k}$ (see Equation 1). All other models in this CV-fold are deleted.

We can use this model selection scheme in two ways. If we have no idea or prior knowledge, which classification or regression method should be used to cope with a specific problem we could use this scheme to look for an empirical answer. In a recent study, we were able to show the superiority of regression trees for predicting the aqueous solubility of small organic compounds, an insight that perfectly correlates with the "chemical intuition" of the experts, working in this field for decades.

The other way is the estimation of model parameters for the different model classes described in Section V.

IV. Data Preprocessing

In some cases it is useful to apply a kind of data preprocessing. All data sets were normalized in the sense that we substracted the mean from the features and divided them by their variance.

In some cases we further balanced the data. If the distribution of the two classes differ in the sense, that one class is only represented with a small number of examples then balancing can improve the the convergence of several training algorithms and has also an impact on the classification error [22]. We apply balancing in the way that we reduce the number of samples in the one class until we have an balanced ratio of the class labels. This will reduce the number of training samples in each CV-fold, hence we have to enlarge the number of CV-folds in order to make use of all available samples. This could be forced by drawing CV-folds with a small overlap.

V. Classification Models

In this section we give a short overview of the models that we use for ensemble building. All models belong to the canonical collection of machine learning algorithms for classification and regressions so details can be found in the references. The implementation of these models[1] in an open source toolbox together with a more detailed description can be found in [23].

A. Linear Discriminant Analysis

The Linear Discriminant Analysis (LDA) is a simple but useful classifier. If we assume that the two classes $k = \{0, 1\}$ have a Gaussian distribution with mean μ_k and they share the same covariance matrix Σ, then the *'linear discriminant function'* $\delta_k(\mathbf{x}), k = \{0, 1\}$ is given by

$$\delta_k(\mathbf{x}) = \mathbf{x}^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + log(\pi_k),$$

where π_k denotes the frequency of occurrence of the class labels. The predicted class labels are given by

$$f(\mathbf{x}) = \text{argmax}_{k=(0,1)} \left\{ \delta_k(\mathbf{x}) \right\} \ .$$

We also implemented two modifications: The Quadratic Discriminant Analysis (QDA) and the Penalized Discriminant Analysis (PDA) as described in detail in Hastie et. al [14]. The parameters of the letter two models are dealing with the penalizing terms.

B. Linear Ridge Model

The linear ridge model is a simple multivariate linear regression that takes the N features $\{\mathbf{x}_i\}_{i=1,\ldots,N}$ as input and the labels $\{y_i\}_{i=1,\ldots,N}$ as output variables while introducing a penalty term λ to the regression coefficients $\alpha = (\alpha_1, \ldots, \alpha_d)$. The regression coefficients minimize a penalized residual sum of squares

$$\alpha = \text{argmin}_\alpha \left\{ \sum_{i=1}^{N} \left(y_i - \alpha_0 - \langle \mathbf{x}_i | \alpha \rangle \right)^2 + \lambda \sum_{j=0}^{d} \alpha_j^2 \right\},$$

where α_0 denotes the constant term in the regression and $\langle \cdot | \cdot \rangle$ is the scalar product defined as

$$\langle \mathbf{x} | \alpha \rangle = \sum_{k=1}^{d} x_k \alpha_k \ .$$

The *'linear discriminant function'* is given by

$$f(\mathbf{x}) = sign \left(\alpha_0 + \langle \mathbf{x} | \alpha \rangle \right) \ .$$

The free parameter of the model is the ridge factor λ.

[1]The toolbox is an open source MATLAB Toolbox that allows the integration of existing implementations of classification algorithms and it contains more then the described model classes.

C. Nearest Neighbor Classifier

A k-Nearest-Neighbor Classifier takes a weighted average over the labels z_i of those observations $\mathbf{z}_i$ in the training set that are closest to the query point $\mathbf{x}$. This denotes as

$$f(\mathbf{x}) = \frac{1}{\sum w_i} \sum_{\mathbf{z}_i \in N_k(\mathbf{x})} w_i z_i,$$

where $N_k(\mathbf{x})$ denotes the k-element neighborhood of $\mathbf{x}$, defined in a given metric and w_i is the related distance. Common choices are the L_1, L_2 and the L_∞ metrics. The parameters of the model are the number of neighbors and the choice of the metric.

D. Trees

Trees are conceptually simple but powerful tools for classification and regression. For our purpose we use the 'classification and regression trees' (CART) as described in Breiman et al. [24]. The main feature of the CART algorithm is the binary decision role that is introduced at each tree node with respect to the information content of the split. In this way the most discriminating binary splits are near the tree root building an hierarchical decision scheme. It is known, that trees have a high variance, so they benefit from the ensemble approach [9]. The parameters of the tree models are related to splitting the tree nodes (the impurity measure and the split criterion, see [14] for a detailed description).

E. Neural Networks

We use a multilayer feed-forward Neural Network (MLP: Multi Layer Perceptron) with the $\tanh(\mathbf{x})$ as activation function. The weights are initialized with Gaussian distributed random numbers having zero mean and scaled variances, following a suggestion of LeCun et al. [25]. The weights are trained with a gradient descend based on the Rprop Algorithm [26] with the improvements given in [27]. The MLP works with a common weight decay with the penalty term

$$P(\vec{w}) = \lambda \sum_{i=1}^{N} \frac{w_i^2}{1 + w_i^2} \ ,$$

where $\vec{w}$ denotes the N-dimensional weight vector of the MLP and a small regularization parameter λ. The number of hidden layers, the number of neurons and the regularization parameter are adjusted during the CV-training.

F. Support Vector Machines

Over the last decade Support Vector Machines (SVMs) have become very powerful tools in machine learning. A SVM creates a hyperplane in a 'feature space' that separates the data into two classes with the maximum-margin. The 'feature space' can be a mapping of the original features $(\mathbf{x}, \mathbf{x}')$ into a higher dimensional space using a positive semi-definite function

$$(\mathbf{x}, \mathbf{x}') \mapsto k(\mathbf{x}, \mathbf{x}').$$

The function $k(\cdot, \cdot)$ is called the *kernel function* and the so called 'kernel trick' uses Mercer's condition, which states that any positive semi-definite kernel $k(\mathbf{x}, \mathbf{x}')$ can be expressed as a dot product in a high-dimensional space (see [28] for a detailed introduction). The theoretical foundations of this approach were given by Vapnik's *Statistical Learning Theory* [29], [30] and later extended to the nonlinear case [31]. We use an implementation of SVMs that is based on the libsvm provided by Chih-Jen Lin [32] with the standard kernels:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= (\mathbf{x} \cdot \mathbf{x}') && \text{linear} \\ &= (\mathbf{x} \cdot \mathbf{x}' + 1)^d && \text{polynomial} \\ &= \exp\left(-\frac{||\mathbf{x}-\mathbf{x}'||}{\sigma^2}\right) && \text{rbf} \end{aligned}$$

The parameters of the model are with respect to the kernel-type the polynomial degree d , the width of the rbf σ^2 and the value concerning the cost of constrain violation during the SVM training.

TABLE I

PREPROCESSING, BASE MODELS AND PARAMETER SETTINGS FOR THE *agnostic learning track*.

Data set	ADA	GINA	HIVA	NOVA	SYLVA
Features	48	970	1617	16969	216
Examples	4147	3153	3845	1754	13086
Normalizing	yes	yes	yes	yes	yes
Balancing	–	–	yes	yes	yes
CV-folds	51	51	51	21	101
R	7/3	1/1	1/1	1/1	1/1
Base models	All listed	CART	CART	CART	All listed

VI. Application to the Agnostic Learning vs. Prior Knowledge Challenge

The *Agnostic Learning vs. Prior Knowledge Challenge* [33] provides five data sets from different sources. The aim of the challenge is to to obtain the best possible predictor on five classification problems using either the raw data or the preprocessed data. We took the *agnostic learning track* where we had to use the preprocessed data. In this section we will report, which preprocessing, parameter settings and base models were used to build classifier ensembles for the 5 agnostic learning data sets (ADA, GINA, HIVA, NOVA and SYLVA, see Table I). In the case of ADA and SYLVA we used all model classes described in Section (V) and performed an extensive search in the space of model parameters. For the three remaining cases, we use only CART models because they delivered the best trade-off concerning time-consumption during the training and performance on the test data sets. In four cases we used a train-test-ratio of 1/1 in order to prevent overfitting effects.

VII. Results and Conclusions

The competition results with respect to the first submission deadline were published on the challenge website [33]. The ranking of our model with respect to the individual data sets is listed in Table I. In overall ranking we reached the 6st position in the field of competitors. The best model

TABLE II

THE RANKING OF OUR MODELS FOR THE *agnostic learning track*.

Data set	ADA	GINA	HIVA	NOVA	SYLVA
Rank	3	8	6	11	2
Model	CART	CART	CART	CART	CART

we used was an ensemble of trees for all five data sets with the parameter settings listed in Table II. The success of tree based classification methods provided by R. Lutz [34] in the WCCI Performance Prediction Challenge in 2006 [35] encouraged us to focus more on CART models and the results show that this was a good decision. We further improved our approach reported in [21] where we focused to much on the performance on the validation set that was provided on the challenge web-site. It turned out, that this was a mistake, because we were trapped by overfitting. This time we used the outcome of the cross-validation to guide the model selection process and we further used a ratio of $R = 1/1$ for the training/test splits (see Equ. 7 and Table I).

ACKNOWLEDGMENT

The author like to thank the colleagues from the Medical Informatics Department at Charité and the Molecular Modelling Group at FMP, in particular Dr. Ronald Kühne for his support and assistance in turbulent times.

REFERENCES

[1] L. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. on Pattern Analysis and Machine Intelligence 12 (10) (1990) 993–1001.

[2] S. Geman, E. Bienenstock, R. Doursat, Neural networks and the bias/variance dilemma, Neural Computation 4 (1992) 1–58.

[3] M. P. Perrone, L. N. Cooper, When Networks Disagree: Ensemble Methods for Hybrid Neural Networks, in: R. J. Mammone (Ed.), Neural Networks for Speech and Image Processing, Chapman-Hall, 1993, pp. 126–142.

[4] R. Kohavi, D. Wolpert, Bias plus variance decomposition for zero-one loss functions, in: L. Saitta (Ed.), Proceedings of the Thirteenth International Conference on Machine Learning, Morgan Kaufmann, 1996, pp. 275–283.

[5] P. Domingos, A unified bias-variance decomposition for zero-one and squared loss, in: Proc. of the 17 th National Conference on Artifcial Intelligence, 2000, pp. 564–569.

[6] A. Krogh, J. Vedelsby, Neural network ensembles, cross validation, and active learning, in: G. Tesauro, D. Touretzky, T. Leen (Eds.), Advances in Neural Information Processing Systems, Vol. 7, The MIT Press, 1995, pp. 231–238.

[7] U. Naftaly, N. Intrator, D. Horn, Optimal ensemble averaging of neural networks, Network, Comp. Neural Sys. 8 (1997) 283–296.

[8] A. Krogh, P. Sollich, Statistical mechanics of ensemble learning, Physical Review E 55 (1) (1997) 811–825.

[9] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.

[10] A. Rothfuss, T. Steger-Hartmann, N. Heinrich, J. Wichard, Computational prediction of the chromosome-damaging potential of chemicals, Chemical Research in Toxicology 19 (10) (2006) 1313–1319.

[11] T. Taskaya-Temizel, M. Casey, A comparative study of autoregressive neural network hybrids, Neural Networks 18 (5-6) (2005) 781–789.

[12] T. Hashem, B. Schmeiser, Improving model accuracy using optimal linear combinations of trained neural networks, IEEE Trans. Neural Networks 6 (3) (1995) 792–794.

[13] E. Kong, T. Dietterich, Error-correcting output coding corrects bias and variance, in: Proc. of the International Conference on Machine Learning, 1995, pp. 313–321.

[14] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer Series in Statistics, Springer-Verlag, 2001.

[15] C. Schaffer, Selecting a classification method by mross-validation, in: Fourth Intl. Workshop on Artificial Intelligence & Statistics, 1993, pp. 15–25.

[16] J. Quinlan, Comparing connectionist and symbolic learning methods, Vol. I, MIT Press, 1994, pp. 445–456.

[17] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (2003) 1157–1182.

[18] D. Wolpert, Stacked generalization, Neural Networks 5 (1992) 241–259.

[19] J. Wichard, M. Ogorzałek, C. Merkwirth, Detecting correlation in stock markets, Physica A 344 (2004) 308–311.

[20] J. Wichard, M. Ogorzałek, Time series prediction with ensemble models, in: Proc. IJCNN 2004, 2004, pp. 1625–1629.

[21] J. Wichard, Model selection in an ensemble framework, in: Proceedings of the IEEE World Congress on Computational Intelligence, Vancouver, Canada, 2006, pp. 4189–4194.

[22] C. Merkwirth, M. Ogorzałek, J. Wichard, Stochastic gradient descent training of ensembles of DT-CNN classifiers for digit recognition, in: Proceedings of the ECCTD, Vol. 2, Kraków, Poland, 2003, pp. 337–341.

[23] J. Wichard, C. Merkwirth, ENTOOL - A Matlab toolbox for ensemble modeling (2007).
URL `http://www.j-wichard.de/entool/`

[24] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wadsworth and Brooks, Monterey, CA, 1984, new edition 1993.

[25] Y. LeCun, L. Bottou, G. Orr, K. Müller, Efficient BackProp, in: G. Orr, K. Müller (Eds.), Neural Networks: Tricks of the trade, Vol. 1524 of Lecture Notes in Computer Science, Springer Verlag, 1998, pp. 9–50.

[26] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: The RPROP algorithm, in: Proc. of the IEEE Int. Conf. on Neural Networks, San Francisco, CA, 1993, pp. 586–591.

[27] C. Igel, M. Hüsken, Improving the Rprop learning algorithm, in: H. Bothe, R. Rojas (Eds.), Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000), ICSC Academic Press, 2000, pp. 115–121.

[28] N. Cristianini, J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods, Cambridge University Press, 2000.

[29] V. Vapnik, A. J. Tscherwonenkis, Theorie der Zeichenerkennung, Akademie Verlag Verlag, Berlin, 1979.

[30] V. Vapnik, The Nature of Statistical Learning Theory, Springer Verlag, New York, 1999.

[31] B. Boser, I. Guyon, V. Vapnik, A training algorithm for optimal margin classifiers, in: In Proc. of the Fifth Annual Workshop on Computational Learning Theory, 1992, pp. 144–152.

[32] C. C. Chang, C. Lin, Libsvm - A library for support vector machines (2001).
URL `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

[33] I. Guyon, et al., Agnostic Learning vs. Prior Knowledge Challenge & Data Representation Discovery Workshop, IJCNN 2007, Orlando, Florida.

[34] R. Lutz, Logitboost with trees applied to the WCCI 2006 performance prediction challenge datasets, in: Proceedings of the IEEE World Congress on Computational Intelligence, Vancouver, Canada, 2006, pp. 2966–2969.

[35] I. Guyon, et al., Performance Prediction Challenge, WCCI 2006, Vancouver, Canada.