Model Selection in an Ensemble Framework

Jörg D. Wichard

***Abstract*— We like to present a method to build ensemble models based on an extended cross-validation approach. The cross-validation puts several model classes in a tournament and selects the best performing model with respect to the validation set. This leads to a model selection strategy and an estimation of the expected modelling error.**

I. Introduction

If we average the output of several different models, we call this an ensemble model. Building ensembles of models is a common way to improve classification and regression models in terms of stability and classification accuracy 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 straight forward 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. One way to achieve diverse models is the well known boostrap aggregating or 'bagging' (see Breiman [9]) where the models are trained on different subsets of the training data. A different way to introduce diversity are heterogeneous ensembles, that consist of several different model classes like Neural Networks, nearest-neighbor models, decision trees, etc [10].

Let us 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.

In the next sections we will present our ensemble approach, the model selection scheme and the method we used to estimate the performance of our classifier ensemble regarding the five classification tasks of the 'Performance Prediction Challenge'.

Jörg D. Wichard is with the Schering AG, Berlin and the Institute of Molecular Pharmacology Molecular Modelling Group, Robert Rössle Straße 10, D-13125 Berlin-Buch, 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. [11]). 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. 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 [12], Kohavi [4] or Domingos [5] to mention a few.

III. Cross Validation

In order to select models for the final ensemble we use cross validation (CV) for model training, model selection and the estimation of the expected classification error. The models are initialized with different model parameters and cross validation helps us to find proper values for these parameters and to select the best performing models for the final ensemble.

A. Data Partitioning

First of all we isolate a *'test set'* that is hold out from the training procedure and only used for the final evaluation (usually 10% to 25% of the entire data set).

For a k-fold CV the data is divided k-times into a *'training set'* and a *'validation set'* (see Figure 1), both sets containing randomly drawn subsets of the data without replications. There is no 'golden rule' concerning the ratio

$$R_{t/v} = \frac{\text{\# training samples}}{\text{\# validation samples}} \tag{2}$$

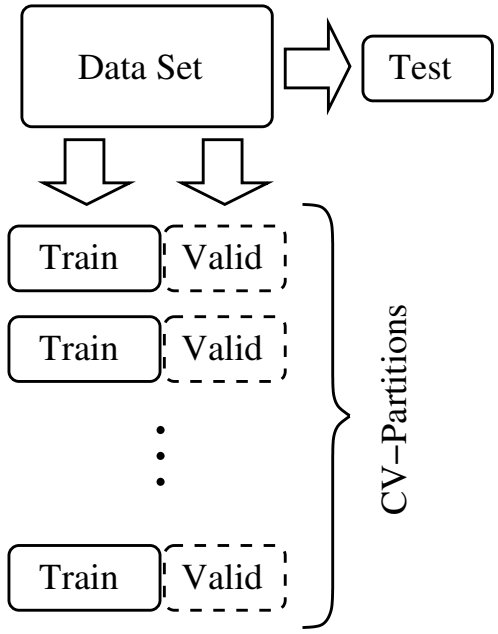

Fig. 1. For every partition of the cross-validation, the data is divided in a training and a validation set. A separate test set is selected once and kept back during the whole training procedure.

but values ranging from $\frac{80}{20}$ up to $\frac{50}{50}$ for $R_{t/v}$ seem to be reasonable for all practical purpose. If the data needs to be balanced, then balancing is only introduced to the training set, the validation and the test set should have the same distribution as the original data, in order to use the classification error on the validation set as an estimation of the expected classification error (see Section VI).

B. Model Selection

After the data partitioning we have k CV-partitions with a training and a validation set each. This leads to k training-validation rounds and in every round we select only one model to become a member of the final ensemble (namely the best model with respect to the validation set). In every round we train several different models classes with a variety of model parameters (see Section IV for an overview of the models and the related model parameters). 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 applied this approach to several regression problems related to time series prediction and we achieved convincing results [10], [13], [14].

In the 'Performance Prediction Challenge' we have refined the strategy slightly. In a first CV training we look for the dominating model classes and possible preprocessing strategies like feature selection, scaling and balancing. In the final CV training, we initialize the base models with a broad variety of model parameter and perform an exhaustive search of the parameter space.

C. Expected Classification Error

We estimate expected classification error $\hat{E}$ from the classification errors on the k validation sets that we used to select the models. If E^k_{valid} denotes the error on the validation set of the best performing model in the k-th round of the CV, then

$$\hat{E} = \frac{1}{k}\sum_{n=1}^{k} E^k_{valid} \tag{3}$$

is the expected classification error with the standard deviation $\sigma(\hat{E})$. We are aware of the fact, that this is a quite optimistic guess, because we use only the errors of the best performing models. For this reason we kept a test set out of the entire training. The test set contains only unseen data points and marks real 'out of training' data. We start the CV training with a ratio $R_{t/v} = \frac{80}{20}$ and calculate the classification error of the ensemble model on the test set E_{test}. If E_{test} lies in the interval $\hat{E} \pm \sigma(\hat{E})$ then CV training is completed. Otherwise we choose a smaller ratio $R_{t/v}$ and start the whole procedure from the beginning unless $R_{t/v} = \frac{50}{50}$. In this case the training and the validation set have the same size and a further shrinkage of the training set usually increases the error on the validation set. During this process we assume, that the test set and the k validation sets have the same properties as the entire data set. We are aware of the fact, that a strong bias in the test set would confuse the whole procedure. Therefore we have to use this method with care. If E_{test} is not located in the interval $\hat{E} \pm \sigma(\hat{E})$ we could draw a new and maybe larger test set. Hastie et al. [15] propose a ratio of 50% for training, 25% for validation and 25% for test as a *rule of thumb*[1].

IV. Classification Models

In this section we give a short overview of the models that we use for ensemble building and the model parameters (hyperparameters) that are optimized in during the CV. All models belong to the classical collection of machine learning algorithms for classification and regression. Therefor we will not go into greater detail that can be found in the references. The implementation of these models in an open source toolbox together with a more detailed description can be found in [16].

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

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

[1] *"It is difficult to give a general rule on how to choose the number of observations in each of the three parts, as it depends on the signal-to-noise ratio in the data and the training sample size. A typical split might be* 50% *for training, and* 25% *each for validation testing."*, see Hastie et al. [15].

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 [17]. 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.

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. [18]. 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 [17] 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. [19]. The weights are trained with a gradient descend based on the Rprop Algorithm [20] with the improvements given in [21]. 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 [22] for a detailed introduction). The theoretical foundations of this approach were given by Vapnik's *Statistical Learning Theory* [23], [24] and later extended to the nonlinear case [25]. We use an implementation of SVMs that is based on the libsvm provided by Chih-Jen Lin [26] with the standart 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.

G. Boosting

The idea in the early days of boosting was to combine the output of many 'weak' classifier to achieve a 'strong committee' with respect to the classification error (see Freud et al. [27] for an overview). But in principle, boosting can also be used to improve the performance of almost any classification algorithm. In our approach we used the *Adaboost.M1* boosting scheme as described by Friedman et al. [28] (see Hastie et al. [17] for a detailed overview and

description) and applied it to the ridge model in Section IV-B. So the base model of the ensemble is a boosted linear ridge model.

V. Data Preprocessing

In some cases it is usefull to apply a kind of data preprocessing in order to select the most discriminant features and reduce the dimensionality of the leaning problem, in particular if we have more features than training examples.

A. Normalizing

If we substract the mean from the features and divide them with their variance, we call this normalizing the data.

B. Balancing

If the distribution of the two classes differ in the sense, that one class is only represented with a small number of examples then we can balance the data in the training set. This can improve the the convergence of several training algorithms and has also an impact on the classification error [29]. 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.

C. PCA

The principal component analysis (PCA) is a linear transformation of the feature space that chooses a new coordinate system for feature data in a way, that the greatest variance by any projection of the features lies on the first new coordinate axis (the first principal component), the second greatest variance on the second principal component, and so on. The principal components forming a new orthonormal basis of the feature space, which can be use to reduce dimensionality while retaining only those characteristics of the features that contribute most to its variance by eliminating the later principal components. In this way, the PCA helps a lot to get a fist impression about the 'real' dimensionality of the problem.

D. Feature Selection with SVMs

There are a lot of known methods to select the relevant features in a given data set (see [30] for an overview). Some are based on simple correlation analysis of the feature values and the class labels, others are simple brute force attacks that enlarge the features iteratively by selection only those that improve the classifier performance. The feature selection algorithm proposed by Guyon et al. [31] is based on the idea to use the weights of a classifier to achieve a ranking of the feature importance. For a detailed description see [31] and the references therein. We found this one of the best performing feature selection algorithms when applied to the data sets in the challenge.

		Predicted	
		Class 1	Class –1
Real	Class 1	tp	fn
	Class –1	fp	tn

Fig. 2. The confusion matrix for a binary classification problem. The notation: tp = 'true positive'; tn = 'true negative'; fp = 'false positive'; fn = 'false negative'.

VI. Error Measures

If we talk about proper classification models, we talk about models with respect to the costs of missclassification. In Figure 2 we show the confusion matrix for a binary classification task. In general the *accuracy*

$$\text{accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

seems to be the 'natural choice' of an error measure for most of the classification problems if the data set is balanced. Another choice is the balanced error rate (BER), defined as

$$\text{BER} = 0.5 * \left(\frac{fp}{tn + fp} + \frac{fn}{fn + tp} \right) . \quad (4)$$

But in several applications these are not the values of interest. If we consider the case, where we have to classify if a person has a particular disease or not, given the features from a medical investigation (a test of the patients blood, for instance), then we want to find all the patients that have this disease with a high probability, paying the price to get some healthy people that have to be tested again with more expansive methods than a simple test of the patients blood. In this case the recall (or 'true positive rate') is the error measure that matters. Some of the models listed in Section IV can handle the problem of different missclassification costs (like the CART model), others cannot. With our approach we are able to select models, that perform well on a given error measure and build classification models for several demands.

VII. Application to the Performance Prediction Challenge

The *Performance Prediction Challenge* [32] provides five data sets from different sources divided in a training set, a validation set and a test set each. The aim of the challenge is to estimate how accurately a given predictive model will perform on test data with respect to the BER defined in Equation 4. In this section we will report, which preprocessing, parameter settings and base models were used to build classifier ensembles for the 5 data sets (ADA, GINA, HIVA, NOVA and SYLVA, see Table VII). In all cases we used the validation set, that was provided on the challenge web-site as our 'hold out test set' (see Section III). This 'test set' was 10% of the size of the training set in all cases. The results of our best challange entry are reported in Table II. In the overall ranking of the *Performance Prediction Challenge* we

TABLE I

FEATURE SELECTION, PREPROCESSING AND BASE MODELS.

Data set	ADA	GINA	HIVA	NOVA	SYLVA
Features	48	970	1617	16969	216
Examples	4147	3153	3845	1754	13086
Positive	1029	1550	135	499	805
Negative	3118	1603	3710	1255	12281
Normalizing	yes	yes	yes	–	–
Balancing	yes	–	yes	–	yes
Feat. select.	–	svm	–	–	–
CV-folds	21	5	11	9	41
$R_{t/v}$	1/1	4/1	3/1	3/1	1/1
Base models	Boosted ridge	SVM (RBF)	Boosted ridge	Boosted ridge	CART

are on the 10th position with respect to the test score and we gained an average ranking of 16.6 as computed by the organizers.

A. ADA

The *ADA* data consist of 1029 positive and 3118 negative samples with 48 features each. We balanced and normalized the data and trained our ensemble with a fraction $R_{t/v} = 1/1$ (see Equ. 2) and $k = 21$ CV-folds. The finale ensemble consisits of boosted ridge models with different ridge parameters λ which where determined during the training.

B. GINA

The *GINA* data consist of 1550 positive and 1603 negative samples with 970 features each. We normalized the data and performed a featere selection based on the method described in Guyon et al. [31] with 450 features remaining in the training set. During the CV-training the SVM with rbf-kernels showed superior performance. We build an SVM classifier ensemble with $k = 5$ CV-folds where the model parameters were adjusted during the training.

C. HIVA

The *HIVA* data consist of 135 positive and 3710 negative samples with 1617 features each. We normalized and balanced the data and build and ensemble model with $k = 11$ CV-folds consiting of boosted ridge models with several ridge parameters and a ratio $R_{t/v} = 3/1$. We didn't find a way to apply feature selection and to introduce a more sophisticated model instead of the simple ridge. It should be stated, that the penalizing term in the ridge model shrinks the 'useless' coefficients to zero, hence operating as an *intrinsic feature selection*.

D. NOVA

The *NOVA* data consist of 499 positive and 1754 negative samples with 16969 features each. We finally run a 9-fold boosted ridge model with an ratio $R_{t/v} = 3/1$ and the ridge parameters adjusted during the training.

TABLE II

RESULTS OF OUR BEST CHALLENGE ENTRY.

Dataset	Test AUC	Test BER	BER guess	Guess error	Test score
ADA	0.8614	0.1801	0.1636	0.0165	0.1965
GINA	0.9866	0.0523	0.05	0.0023	0.0543
HIVA	0.7172	0.3057	0.338	0.0323	0.3377
NOVA	0.9459	0.0611	0.08	0.0189	0.08
SYLVA	0.9956	0.0267	0.007	0.0197	0.0464
Overall	0.9013	0.1252	0.1277	0.0179	0.143

E. SYLVA

The *SYLVA* data consist of 805 positive and 12281 negative samples with 216 features each. We balanced the data and tested several model classes, at least the CART models were the winners. We trained a ensemble with $k = 41$ CV-folds and and fraction $R_{t/v} = 1/1$ using only trees with no pruning to achieve a high model variance. It is known, that ensembles of trees benefit from a high number of ensemble members, so we decided to choose 41 CV-folds which leads to 41 trees in the final ensemble.

VIII. CONCLUSIONS

We showed that the ensemble approach described above works a method to select classification models and to estimate the expected classification error. The fair competition of models - trained and validated on the same data sets - doesn't prefer any model by preliminary choice. Only the performance on the validation set is the criterion to be selected as an ensemble member.

After taking a look at the over all ranking and after analyzing our earlier submissions we have to admit, that we focused to much on the 10% test set during the model selection. We introduced a bias in our appraoch that we couldn't handle. This seems to be the classical overfitting trap.

The *Performance Prediction Challenge* showed in our eyes, that there is no perfect black box model that works in any case right *'from the shelf'*. Every modelling or classification task has its own challenge and needs a careful parameter adjustment, a feature selection strategy and at least a fundamental knowledge about the underlining algorithms.

ACKNOWLEDGMENTS

The author like to thank all colleagues from the Molecular Modelling Group at FMP and the from the Computational Chemistry Department at Schering for support and fruitful discussions.

REFERENCES

[1] L. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

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

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

[4] R. Kohavi and D. Wolpert, "Bias plus variance decomposition for zero-one loss functions," in *Proceedings of the Thirteenth International Conference on Machine Learning*, L. Saitta, Ed. 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 Inteligence*, 2000, pp. 564–569.

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

[7] U. Naftaly, N. Intrator, and D. Horn, "Optimal ensemble averaging of neural networks," *Network, Comp. Neural Sys.*, vol. 8, pp. 283–296, 1997.

[8] A. Krogh and P. Sollich, "Statistical mechanics of ensemble learning," *Physical Review E*, vol. 55, no. 1, pp. 811–825, 1997.

[9] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[10] J. Wichard and M. Ogorzałek, "Time series prediction with ensemble models applied to the cats benchmark," *Neurocomputing*, 2006, to appear in a special issue of Neurocomputing.

[11] T. Hashem, B. Schmeiser, and Y. Yih, "Optimal linear combinations of neural networks: An overview," in *Proc. IEEE Int. Conf. on Neural Networks*, 1994.

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

[13] J. Wichard, M. Ogorzałek, and C. Merkwirth, "Detecting correlation in stock markets," *Physica A*, vol. 344, pp. 308–311, 2004.

[14] J. Wichard and M. Ogorzałek, "Iterated time series prediction with ensemble models," in *Proceedings of the 23rd International Conference on Modelling Identifi cation and Control*, 2004.

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. Springer-Verlag, 2001, ch. 7.2, p. 196.

[16] C. Merkwirth and J. Wichard, "ENTOOL - A Matlab toolbox for ensemble modeling," 2002. [Online]. Available: http://chopin.zet.agh.edu.pl/˜wichtel/

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

[18] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classifi cation and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984, new edition 1993.

[19] Y. LeCun, L. Bottou, G. Orr, and K. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the trade*, ser. Lecture Notes in Computer Science, G. Orr and K. Müller, Eds. Springer Verlag, 1998, vol. 1524, pp. 9–50.

[20] M. Riedmiller and 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.

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

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

[23] V. Vapnik and A. J. Tscherwonenkis, *Theorie der Zeichenerkennung*. Berlin: Akademie Verlag Verlag, 1979.

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

[25] B. Boser, I. Guyon, and 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.

[26] C. C. Chang and C. Lin, "Libsvm - A library for support vector machines," 2001. [Online]. Available: http://www.csie.ntu.edu.tw/˜cjlin/libsvm

[27] Y. Freund and R. Schapire, "A short introduction to boosting," *J. Japan. Soc. for Artif. Intel.*, vol. 14(5), pp. 771–780, 1999.

[28] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.

[29] C. Merkwirth, M. Ogorzałek, and 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.

[30] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[31] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[32] I. Guyon and et al., "Performance Prediction Challenge, WCCI 2006, Vancouver, Canada."