Building Ensembles with Heterogeneous Models

Jörg D. Wichard[1], Christian Merkwirth[2] and Maciej Ogorzałek[1]

[1] AGH University of Science and Technology
Department of Electrical Engineering
al. Mickiewicza 30
30-059 Kraków, Poland
JoergWichard@web.de maciej@agh.edu.pl
[2] Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany
ChristianMerkwirth@web.de

Abstract. In the context of ensemble learning for regression problems, we study the effect of building ensembles from different model classes. Tests on real and simulated data sets show that this approach can improve model accuracy compared to ensembles from a single model class.

1 Introduction

Ensemble building is a common way to improve the performance of the resulting model for classification and regression tasks, since it was noticed that an ensemble of individual predictors performs better than a single predictor in the average. Almost all ensemble methods described so far use models of one single class, e.g. neural networks [10, 17, 14, 16] or regression trees [3].
We suggest to build ensembles of different model classes to improve the ensemble performance in regression problems. The theoretical background of our approach is provided by the bias/variance decomposition of the generalization error given in the next section. We argue that an ensemble of heterogeneous models leads to a reduction of the ensemble variance because the errors of the individual components have small correlation and thus the cross terms in the variance are small. The used models and the training procedures are described in section 3 and 4. In section 5 we show numerical simulations on different data sets.

2 The Bias/Variance Decomposition for Ensembles

Our approach is based on the observation that the generalization error of an ensemble model could be improved if the predictors on which averaging is done disagree and if their fluctuations are uncorrelated [13]. 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 future observations of $\mathbf{x}$. These future 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

$$Err(\mathbf{x}, D) = E[(y - f(\mathbf{x}))^2 | \mathbf{x}, D], \tag{1}$$

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

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

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 [9] the bias/variance decomposition of $Err(\mathbf{x})$ is

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

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 [11]. We now consider the case of an ensemble average $\hat{f}(\mathbf{x})$ consisting of K individual models

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{K} \omega_i f_i(\mathbf{x}), \tag{3}$$

where the weights may sum to one $\sum_{i=1}^{K} \omega_i = 1$. If we put this into equ. (2) we get

$$Err(\mathbf{x}) = \sigma^2 + Bias(\hat{f}(x))^2 + Var(\hat{f}(x)), \tag{4}$$

and we can have a look at the effects concerning bias and variance. The bias term in equ. (4) 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:

$$\begin{aligned} Var(\hat{f}) &= E\left[(\hat{f} - E[\hat{f}])^2\right] \\ &= E[(\sum_{i=1}^{K} \omega_i f_i)^2] - (E[\sum_{i=1}^{K} \omega_i f_i])^2 \\ &= \sum_{i=1}^{K} \omega_i^2 \left(E\left[f_i^2\right] - E^2\left[f_i\right]\right) + 2 \sum_{i<j} \omega_i \omega_j \left(E\left[f_i f_j\right] - E\left[f_i\right] E\left[f_j\right]\right), \end{aligned} \tag{5}$$

where the expectation is taken with respect to D. The first sum in equ. (5) 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 [13]. That shows that the reduction in the variance of the ensemble is related to the degree of independence of the single models [16]. This is the key feature of the ensemble approach. There are several ways to introduce model diversity. In the case of neural network ensembles, the networks can have different architecture, different training algorithms or different training subsets [17, 14]. For fixed architectures it is sufficient to use different initial conditions for the training [16]. Another way of variance reduction is Bagging[3], where an ensemble of predictors is trained on several bootstrap replicates of the training set [3]. In Nearest-Neighbor Models the number of neighbors and the metric coefficients could be used to generate diverse models.
We suggest to introduce a new source of ensemble diversity by averaging different model classes. This is based on the observation that the error terms of models from different classes are less correlated than the errors of models from the same class.

3 Models

In this section we give a short overview about the models that we use for ensemble building and the related training procedures. The implementation of these models together with a more detailed description can be found in [7].

3.1 Linear and Polynomial Models

The d-dimensional linear model has the form $f(\mathbf{x}) = a_0 + \sum_{i=1}^{d} a_i x_i$, where a_0 is the offset. The model is trained on randomly selected subsets of the training data, where 20% of the training set is used for model validation and the best 66% of the validated models are used to build the average linear model. The coefficients $a_{i=0,\ldots,d}$ are calculated with a standard least square method [18].
The polynomial model is given by $f(\mathbf{x}) = \sum_{i=1}^{P} a_i p_i(\mathbf{x})$, wherein the monoms have the form $p_i(\mathbf{x}) = \prod_{i=1}^{d} x_i^{n_i}$. We use an iterative term selection for the monoms wherein we add successively the terms to the polynomial model that decrease the out-of-sample error on a subset of the training data.

3.2 Neural Networks

We use a multilayer feed-forward neural network (MLP: Multi Layer Perceptron) with the $\tanh(\mathbf{x})$ as nonlinear element. In order to increase the ensemble ambiguity, we initialize the weights with Gaussian distributed random numbers having zero mean and scaled variances, following a suggestion of LeCun et al. [15]. The number of hidden layers is chosen at random to be one or two and the

[3] This method was extended by using the unused training examples to reduce bias as well as variance [2].

numbers of neurons in also random (3-9 Neurons in the first layer, 4-32 in second layer). We use two different training procedures. A first order training based on the Rprop Algorithm [19] with the improvements given in [12]. The second order training is a Levenberg Marquart Gradient Descent [15]. As regularization method we use the common weight decay with the penalty term

$$P(\boldsymbol{w}) = \lambda \sum_{i=1}^{N} \frac{w_i^2}{1 + w_i^2} \,, \tag{6}$$

where $\boldsymbol{w}$ denotes the N-dimensional weight vector of the MLP and the regularization parameter is small $\lambda = 0.001$.

3.3 Perceptron Radial Basis Net

Perceptron Radial Basis Net (PRBFN) is and extension of MLP and Radial Basis Function (RBF) Networks. The PRBFN combine RBF and sigmoid units in the hidden layers. This hybrid network architecture together with a sophisticated training procedure was introduced by Cohen and Intrator [4]. The number and the centers of the hidden units are generated dynamically during the training and network parameters are refined by gradient descent, as described in [5].

3.4 Nearest-Neighbor Models

A k-Nearest-Neighbor model takes a weighted average over those observations y_i in the training set that are closest[4] to the query point $\boldsymbol{x}$. This is,

$$f(\boldsymbol{x}) = \frac{1}{\sum w_i} \sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} w_i y_i, \tag{7}$$

where $N_k(\boldsymbol{x})$ denotes the k-element neighborhood of $\boldsymbol{x}$, defined in a given metric. Common choices are the L_1, L_2 and the L_∞ metrics. To compensate for irrelevant input dimensions, distances are computed using a weighted metric:

$$d(\boldsymbol{x}, \boldsymbol{y}) = (\sum_{i=1}^{D} m_i (x_i - y_i)^M)^{\frac{1}{M}} \quad 0 \leq m_i \leq 1. \tag{8}$$

The vector of metric coefficients $\boldsymbol{m}$ is adapted by a Genetic Algorithm. One vector of metric coefficients is an individual of the population. The fitness value is assigned to each individual according to it's error on the training data set. For our investigation we employed the fast nearest neighbor algorithm ATRIA [1].

[4] Self matches of data set points (i.e. each point is considered to be it's own nearest neighbor) are prohibited by default since this would strongly bias the error on the training set.

4 Model Training and Cross Validation

In order to select models for the final ensemble we use a cross validation scheme for model training. Due to the fact that the models are initiated with different model parameters (number of hidden units, number of nearest neighbors, initial weights, etc.), cross validation helps us to find proper values for these model parameters.
The cross validation is done in several training rounds on different subsets of the entire training data. In every training round the data is divided in a training set and a test set. The trained models are compared by evaluating their prediction errors on the (unseen) test set. The model with the smallest test error is chosen as a member of the ensemble. This is repeated several times and the final ensemble is a simple average over its members. For example a K-fold cross validation training leads to an ensemble with K members, where the weights in equ.(3) are $\omega_i = \frac{1}{K}$.

5 Numerical Simulations

We test our approach on different data sets and estimate bias, variance and generalization error with a Monte Carlo simulation as described in [9]. We generate a large data set D of input output pairs $(\mathbf{x}, y)$ for evaluation and 100 training sets $\{D_i\}_{i=1,\ldots,100}$ of fixed size N to train the corresponding ensemble models $f(\mathbf{x}, D_i)$.
Let $\bar{f}(\mathbf{x}) = \frac{1}{100}\sum_{i=1}^{100} f(\mathbf{x}, D_i)$ denote the average of these models evaluated on D, than the squared bias and the variance are estimated as

$$Bias(f(\mathbf{x}))^2 = \left(\bar{f}(\mathbf{x}) - E[y|\mathbf{x}]\right)^2$$
$$Var(f(\mathbf{x})) = \frac{1}{100}\sum_{i=1}^{100}\left(\bar{f}(\mathbf{x}) - f(\mathbf{x}, D_i)\right)^2.$$

On every training set $\{D_i\}$ we train five "pure" and one "mixed" ensemble model applying a five-fold cross validation as described in the previous section:

- MLP 1: Ensemble of Multi-layer Neural Networks trained with a first oder gradient descent
- MLP 2: Ensemble of Single-layer Neural Networks trained with a second oder gradient descent
- PRBFN: Ensemble of Perceptron Radial Basis Nets
- NN: Ensemble of Nearest-Neighbor Models with adaptive metrics
- POLY: Ensemble of Polynomial Regression Models (including linear regression as well)
- MIXED: Ensemble build of the five model types listed above

In every cross validation training round, we train five models with randomized initial conditions and add the best performer to the final ensemble. In order to

have a fair competition, the estimation of bias and variance is calculated for every ensemble on the same data set D. Further we normalized the estimators of bias and variance with the variance of the test data. The results for the different datasets are reported below.

5.1 Peaks Function

The Peaks Function is a smooth two dimensional function defined as

$$y = 3(1-x_1)^2 e^{(-(x_1^2)-(x_2+1)^2)} - 10(\frac{x_1}{5} - x_1^3 - x_2^5)e^{(-x_1^2-x_2^2)} - \frac{1}{3}e^{(-(x_1+1)^2-x_2^2)} + \epsilon.$$

The components are uniformly distributed in the rectangle $-3 \leq \{x_i\}_{i=1,2} \leq 3$ and the noise ϵ is distributed as $N(0, \sigma^2)$. We generated 100 training sets D_i with 200 points and a test set with 1000 points for estimating bias and variance. We ran two simulations, one without noise and a second one with noise, where the variance of the added noise was 10% of the variance of the training data. In the noise free case, the mixed ensemble has the lowest variance and the lowest error. In the the noisy case, the mixed ensemble has still the lowest error but the variance term of the PRBFN is a little lower payed with a higher bias term.

Table 1. Simulation results for the Peaks Function in the noise free case (left) and with 10% noise (right).

Ensemble	Bias^2	Variance	Err	Bias^2	Variance	Err
MLP 1	0.031036	0.029292	0.060328	0.076857	0.11974	0.1966
MLP 2	0.07066	0.032882	0.10354	0.13949	0.12867	0.26816
NN	0.024699	0.053831	0.07853	0.064028	0.12584	0.18986
PRBFN	0.049478	0.035246	0.084725	0.091475	0.066117	0.15759
POLY	0.38869	0.12205	0.51074	0.50238	0.3339	0.83628
MIXED	0.015472	0.024695	0.040167	0.048619	0.071661	0.12028

5.2 Friedman Data

This is an artificial data set, appearing in the MARS paper [8]. The functional dependence is given by

$$y = \sqrt{x_1^2 + (x_2 x_3 - \frac{1}{x_2 x_4})^2} + \epsilon,$$

where ϵ is additive noise distributed as $N(0, \sigma^2)$ and the components of $\mathbf{x}$ are uniformly distributed over the ranges $0 \leq x_1 \leq 100$, $20 \leq \frac{x_2}{2\pi} \leq 280$, $0 \leq x_3 \leq 1$, $1 \leq x_4 \leq 11$. We generated 100 training sets D_i with 200 points and a test set with 1000 points for estimating bias and variance. The mixed ensemble has the lowest error and the lowest variance.

Table 2. Simulation results for the Friedman Data

Ensemble	$Bias^2$	Variance	Err
MLP 1	0.18522	0.074267	0.25949
MLP 2	0.18431	0.073714	0.25803
NN	0.29703	0.087773	0.3848
PRBFN	0.22928	0.065533	0.29481
POLY	0.15655	0.091918	0.24847
MIXED	0.17581	0.063802	0.23961

5.3 Abalone Data

The task is to predict the age of abalone from measurements of eight attributes. More details about the dataset are available from the DELF Database [6]. We

Table 3. Simulation results for the Abalone Data

Ensemble	$Bias^2$	Variance	Err
MLP 1	0.41647	0.09954	0.5160
MLP 2	0.41622	0.18298	0.5992
NN	0.47699	0.06035	0.5373
PRBFN	0.44424	0.04426	0.4885
POLY	0.5226	0.0459	0.5684
MIXED	0.43982	0.041441	0.4813

splited the 4177 points of the original abalone data in a test set with 2000 points and used the the other 2177 points to generate 100 subsets D_i with 200 points for the Monte Carlo simulation.

5.4 Concluding Remarks

Our investigation leads to the point, that building ensembles from a variety of model classes can improve model accuracy compared to the case, where we build "pure" ensembles restricted to a single class. This is an encouraging result, since any model class performing well in a specific setting can be incorporated without degrading performance.

Acknowledgments

The work was done as part of the Research Training Network *COSYC of SENS* No. HPRN-CT-2000-00158 within the 5th EU Framework Program of the European Community. The authors thank Simon Cohen for contributing his PRBFN implementation to our toolbox.

References

1. Merkwirth, C., Parlitz, U., Lauterborn, W.: Fast Exact and Approximate Nearest Neighbor Searching for Nonlinear Signal Processing. Phys. Rev. E **62** (2) (2000) 2089–2097
2. Breiman, L.: Using Adaptive Bagging to Debias Regressions. Tec. rep., Statistics Department, University of California at Berkeley (1999)
3. Breiman, L.: Bagging Predictors. Machine Learning **24** (2) (1996) 123–140
4. Cohen, S., Intrator, N.: A Hybrid Projection Based and Radial Basis Function Architecture. In: Proc. Int. Workshop on Multiple Classifier Systems, (2000) 147–155
5. Cohen, S., Intrator, N.: Automatic Model Selection in a Hybrid Perceptron/Radial Network. In: J. Kittler, F. Roli (Eds.): Multiple Classifier Systems. Springer Lecture Notes in Computer Science Vol. 2096, (2001) 440–454
6. Delve Datasets: `http://www.cs.utoronto.ca/~delve/data/datasets.html`
7. Wichard, J.D., Merkwirth, C.: ENTOOL - A Matlab Toolbox for Ensemble Modeling `http://chopin.zet.agh.edu.pl/~wichtel/`
8. Friedman, J.H.: Multivariate adaptive regression splines. Ann. Statist. **19**, (1991) 1–142
9. Geman, S., Bienenstock, E., Doursat, R.: Neural Networks and the Bias/Variance Dilemma. Neural Computation **4**, (1992) 1–58
10. Hansen, L.K., Salamon, P.: Neural Network Ensembles. IEEE Trans. Pattern Analysis and Machine Intelligence **12** (10) (1990) 993–1001
11. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, Springer Series in Statistics. Springer-Verlag, Berlin Heidelberg New York (2001)
12. Igel, C., Hüsken, M.: Improving the Rprop Learning Algorithm. In: Bothe, H., Rojas, R. (eds.): Proceedings of the Second International ICSC Symposium on Neural Computation. ICSC Academic Press, Berlin (2000) 115–121
13. Krogh, A., Sollich, P.: Statistical Mechanics of Ensemble Learning. Physical Review E **55** (1) (1997) 811–825
14. Krogh, A.Vedelsby, J.: Neural Network Ensembles, Cross Validation, and Active Learning. In: Tesauro, G., Touretzky, D., Leen, D. (eds.): Advances in Neural Information Processing Systems. The MIT Press (1995) 231–238
15. LeCun, Y., Bottou, L., Orr, G., Mller, K.; Efficient BackProp. In: Orr,G., Mller,K. (eds.): Neural Networks: Tricks of the trade. Lecture Notes in Computer Science, Vol. 1524. Springer-Verlag, Berlin Heidelberg New York (1998) 9–50
16. Naftaly, U., Intrator, N., Horn, D.: Optimal Ensemble Averaging of Neural Networks. Network, Comp. Neural Sys. **8** (1997) 283–296
17. Perrone M.P., Cooper, L.N.: When Networks Disagree: Ensemble Methods for Hybrid Neural Networks. In: Mammone R.J. (ed.): Neural Networks for Speech and Image Processing. Chapman-Hall (1993) 126–142
18. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press, Cambridge (1992)
19. Riedmiller, M., Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: Proc. of the IEEE Intl. Conf. on Neural Networks. San Francisco (1993) 586–591