Published as a book chapter in: *Handbook for Neural Network Signal Processing,* Yu Hen Hu and Jenq-Neng Hwang (eds.), CRC Press, 2001.

Committee Machines

Volker Tresp[*]
Siemens AG, Corporate Technology
Otto-Hahn-Ring 6
81730 München, Germany

July 31, 2001

Abstract

In this chapter, we describe some of the most important architectures and algorithms for committee machines. We discuss three reasons for using committee machines. The first reason is that a committee can achieve a test set performance unobtainable by a single committee member. As typical representative approaches, we describe simple averaging, bagging and boosting. Secondly with committee machines, one obtains modular solutions which is sometimes desirable. The prime example here is the mixture of experts approach whose goal it is to autonomously break up a complex prediction task into subtasks which are modeled by the individual committee members. The third reason for using committee machines is a reduction in computational complexity. In the presented Bayesian committee machine, the training data set is partitioned into several smaller data sets and the different committee members are trained on the different sets. Their predictions are then combined using a covariance-based weighting scheme. The computational complexity of the Bayesian committee machine approach grows only linearly with the size of the training data set, independent of the learning systems used as committee members.

1 Introduction

In committee machines, an ensemble of estimators —consisting typically of neural networks or decision trees— is generated by means of a learning process and the prediction of the committee for a new input is generated in form of a combination of the predictions of the individual committee members. Committee machines can be useful in many ways. First, the committee might exhibit a test set performance unobtainable by an individual committee member on its own. The reason is that the errors of the individual committee members cancel out to some degree when their predictions are combined. The surprising discovery of this line of research is that even if the committee members were trained on "disturbed" versions of the *same* data set, the predictions of the individual committee members might be sufficiently different such that this averaging process takes place and is beneficial. This line of research is described in Section 2. A second reason for using committee machines is modularity. It is sometimes beneficial if a mapping from input to target is not approximated by one estimator but by several estimators, where each estimator can focus on a particular region in input space. The prediction of the

[*]e-mail: Volker.Tresp@mchp.siemens.de

committee is obtained by a locally weighted combination of the predictions of the committee members. It could be shown that in some applications the individual members self-organize in a way such that the prediction task is divided into meaningful modules. The most important representatives of this line of research are the mixture of experts approach and its variants which are described in Section 3. The third reason for using committee machines is a reduction in computational complexity. Instead of training one estimator using all training data it is computationally more efficient for some type of estimators to partition the data set into several data sets, train different estimators on the individual data sets and then combine the predictions of the individual estimators. Typical examples of estimators for which this procedure is beneficial are Gaussian process regression, kriging, regularization neural networks, smoothing splines and the support vector machine, since for those systems, training time increases drastically with increasing training data set size. By using a committee machine approach, the computational complexity increases only linearly with the size of the training data set. In Section 4 it is shown how the estimates of the individual committee members can be combined such that the performance of the committee is not substantially degraded compared to the performance of one system trained on all data.

The interest of the machine learning community in committee machines began around the middle of the 90's of the last century and this field of research is still very active. A recent compilation of important work can be found in [1] and two web sites dedicated to the issue of committee machines can be found at http://melone.tussy.uni-wh.de/˜chris/ensemble and http://www.boosting.org. In the addition to the activities in the machine learning community, there has been considerable work on committee machines in the area of economical forecasting [2]. One should also emphasize that traditional Bayesian averaging can be interpreted as a committee machine approach. Assume that a statistical model allows the inference about the variable y in form of the predictive probability density $P(y|w)$ where w is a vector of model parameters. Furthermore let's assume that we have a data set D which contains information about the parameter vector w in form of the probability density $P(w|D)$. We then obtain

$$P(y|D) = \int P(y|w)P(w|D)dw \approx \frac{1}{M}\sum_{i=1}^{M} P(y|w_i),$$

where M samples $\{w_i\}_{i=1}^{M}$ are generated from the distribution $P(w|D)$. The last approximation tells us that for Bayesian inference, one should average the predictions of a committee of estimators. This form of Bayesian averaging will not be discussed further in this book chapter and interested readers should consult the literature on Bayesian statistics, e.g. the book by Bernardo and Smith [3].

2 Averaging, Bagging and Boosting

2.1 Introduction

The basic idea is to train a committee of estimators and combine the individual predictions with the goal of achieving improved generalization performance if compared to the performance achievable with a single estimator. As estimators most researchers either use neural networks or decision trees, generated with CART or C4.5 [4].

In *regression*, the committee prediction for a test input x is achieved by forming a weighted sum of the predictions of the M committee members

$$\hat{t}(x) = \sum_{i=1}^{M} g_i f_i(x)$$

where $f_i(x)$ is the prediction of the i-th committee member at input x and where the g_i are weights which are often required to be positive and to sum to one.

In *classification*, the combination is typically implemented as a voting scheme. The committee assigns the pattern to the class which obtains the majority of the (possibly weighted) vote

$$\widehat{class}(x) = \arg\max_j \sum_{i=1}^{M} g_i f_{i,class=j}(x)$$

where $f_{i,class=j}(x)$ is the output of the classifier i for class j. The output typically either corresponds to the posterior class probability $f_{i,class=j}(x) \in [0,1]$ or to a binary decision $f_{i,class=j}(x) \in \{0,1\}$.

Committee machines can be generalized in various directions. In Sections 3 and 4 we will use non-constant weights, i.e., weighting functions which reflect the relevance of a committee member given the input. Also it might not be immediately obvious, but —as explored in Section 4— predictions of the committee members at other inputs can help to improve the prediction at a given x.

The motivation for pursuing committee methods can be understood by analyzing the prediction error of the combined system which is particularly simple if we use a squared error cost function. The expected squared difference between the prediction of a committee member f_i and the true but unknown target t (for simplicity, we don't denote the dependency on x in most parts of this section explicitly)

$$\begin{aligned} E(f_i - t)^2 &= E(f_i - m_i + m_i - t)^2 \\ &= E(f_i - m_i)^2 + E(m_i - t)^2 + 2E\left((f_i - m_i)(m_i - t)\right) \\ &= var_i + b_i^2 \end{aligned} \tag{1}$$

decomposes into the variance $var_i = E(f_i - m_i)^2$ and the square of the bias $b_i = m_i - t$ with $m_i = E(f_i)$. $E(.)$ stands for the expected value which is calculated with respect to different data sets of the same size and possibly variations in the training procedure such as different initializations of the weights in a neural network. As stated earlier, we are interested in estimating t by forming a linear combination of the f_i

$$\hat{t} = \sum_{i=1}^{M} g_i f_i = g' f$$

where $f = (f_1, \ldots, f_M)'$ is the vector of the predictions of the committee members and where $g = (g_1, \ldots, g_M)'$ is the vector of weights. The expected error of the combined system is [5, 6]

$$\begin{aligned} E(\hat{t} - t)^2 &= E(g'f - E(g'f))^2 + E(E(g'f) - t)^2 \\ &= E(g'(f - E(f)))^2 + E(g'm - t)^2 \\ &= g'\Omega g + (g'm - t)^2 \end{aligned} \tag{2}$$

where Ω is an $M \times M$ covariance matrix with

$$\Omega_{ij} = E[(f_i - m_i)(f_j - m_j)]$$

and where $m = (m_1, \ldots, m_M)'$ is the vector of the expected values of the predictions of the committee members. Here, $g'\Omega g$ is the variance of the committee and $g'm - t$ is the bias of the committee.

If we simply average the predictors, i.e., we set $g_i = 1/M$, the last expression simplifies to

$$E(\hat{t} - t)^2 = \frac{1}{M^2}\sum_{i=1}^{M}\Omega_{ii} + \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1, j\neq i}^{M}\Omega_{ij} + \frac{1}{M^2}\left(\sum_{i=1}^{M}(m_i - t)\right)^2. \tag{3}$$

If we now assume that the mean $m_i = mean$, the variance $\Omega_{ii} = var$ and the inter-member covariances $\Omega_{ij} = cov$ are identical for all members, we obtain

$$E(\hat{t} - t)^2 = \frac{1}{M}var + \frac{M^2 - M}{M^2}cov + (mean - t)^2.$$

It is apparent that the bias of the combined system $(mean - t)$ is identical to the bias of each member and is not reduced. Therefore, estimators should be used which have low bias and regularization —which introduces bias— should be avoided. Secondly, the estimators should have low covariance, since this term in the error function cannot be reduced by increasing M. The good news is that the term which results from the variances of the committee members decreases as $1/M$. Thus, if we have estimators with low bias and low covariance between members, the expected error of the combined system is significantly less than the expected errors of the individual members. So in some sense, a committee can be used to reduce both bias and variance: bias is reduced in the design of the members by using little regularization and variance is reduced by the averaging process which takes place in the committee. Unfortunately, things are not quite as simple in practice and we are faced with another version of the well-known bias-variance dilemma (see also Section 2.3).

In regression, $t(x)$ corresponds to the optimal regression function and $f_i(x)$ to the prediction of the i-th estimator. Here, the squared error is a commonly used and the bias-variance decomposition described in this section is applicable. In (two-class) classification $t(x)$ might correspond to the probability for class one, $1 - t(x)$ to the probability for class two, and $f_i(x)$ is the estimate of the i-th estimator for $t(x)$. Although it is debatable if the squared error is the right error measure for classification, the previously described decompositions are applicable as well. In contrast, if we consider as the output of a classifier a decision, i.e., the assigned class, the previously described bias-variance decomposition cannot be applied. A number of alternative bias-variance decompositions for this case have been described in the literature. In particular, Breiman describes a decomposition in which the role of the variance is played by a term named *spread* [7]. In the same paper the author showed that the spread can dramatically be reduced by bagging, a committee approach introduced in Section 2.3.

2.2 Simple Averaging and Simple Voting

In this approach, committee members are typically neural networks. The neural networks are all trained on the complete training data set. A decorrelation among the neural network predictions is typically achieved by varying the initial conditions in training the neural networks such that different neural networks converge into different local minima of the cost function. Despite its simplicity, this procedure is surprisingly successful and turns an apparent disadvantage, local minima in training neural networks, into something useful. This approach was initialized by the work of Perrone [8] and drew a lot of attention to the concept of committee machines. Using the Cauchy inequality, Perrone could show that even for correlated and biased predictors the squared prediction error of the committee machine (obtained by averaging) is equal or less to the mean squared prediction error of the committee members, i.e.,

$$(\hat{t} - t)^2 \leq \frac{1}{M} \sum_{i=1}^{M} (f_i - t)^2.$$

Loosely speaking, this means that as long as the committee members have good prediction performance, averaging cannot really make things worse; it is as good as the average model or better. This can also be understood from the work of Krogh and Vedelsby [9]. Again applied to the special case of averaging (i.e., $g_i = 1/M$) they show that (using the notation of Section 2.1)

$$(\hat{t} - t)^2 = \frac{1}{M} \sum_{i=1}^{M} (f_i - t)^2 - \frac{1}{M} \sum_{i=1}^{M} (f_i - \hat{t})^2 \tag{4}$$

which means that the generalization error of the committee is equal to the average of the generalization error of the members minus the average variance of the committee members (the ambiguity) which immediately leads to the previous bound. In highly regularized neural networks, the ensemble ambiguity is typically small and the generalization error is essentially equal to the average generalization error of the committee members. If neural neural networks are not strongly regularized the ensemble ambiguity is high and the generalization error of the committee should be much smaller than the average generalization error of the committee members. Note that the last equation is valid for a given committee. The *expected* value of the right side is identical to the right side of Equation 3.

2.3 Bagging

Despite the success of the procedures described in the previous section, it became quickly clear that if the training procedure is disturbed such that the correlation between the estimators is reduced, the generalization performance of the combined systems can further be improved. The most important representative of this approach was introduced by Breiman under the name of bagging (<u>b</u>ootstrap <u>agg</u>regation) [7]. The idea behind bagging can be understood in the following way. Let's assume that each committee member is trained on a different data set. Then, surely, the covariance between the predictions of the individual members is zero. Unfortunately, we typically have to work with a fixed training data set. Although it is then impossible to obtain a different training data set for each member, we can at least mimicry this process by training each member on a bootstrap sample of the original data set. Bootstrap data sets are generated by drawing randomly K data points from the original data set of size K with replacement. This means that some data points will appear more than once in a given new data set and some will not appear at all. We repeat this procedure M times and obtain M non-identical data sets which are then used to train the estimators. The output of the committee is then obtained by simple averaging (regression) or by voting (classification).

Experimental evidence suggests that bagging typically outperforms simple averaging and voting. Breiman makes the point that committee members should be unstable for bagging to work. By unstable it is meant that the estimators should be sensitive to changes in the training data set, e.g. neural networks should not be strongly regularized. But recall that well regularized neural networks in general perform better than under-regularized neural networks and we are faced with another version of the well known bias-variance dilemma: if we use under-regularized neural networks we start with suboptimal committee members but bagging improves performance considerably. If we start with well-regularized neural networks we start with well-performing committee members but bagging does not significantly improve performance. Experimental evidence indicates that the optimal degree of regularization is problem-dependent [6]. Breiman himself mostly works with CART decision trees as committee members.

There are other ways of perturbing the training data set besides using bootstrap samples for training (e.g. adding noise, flipping classification labels) [10, 11] which in some cases also work well. Raviv and Intrator [12] report good results by adding noise on bootstrap samples to increase the decorrelation between committee members, in combination with a sensible regularization of the committee members.

2.4 Boosting

In this section, we will only discuss the classification case. The difference to the previous approaches is that in boosting, committee members are trained sequentially, and the training of a particular committee member is dependent on the training and the performance of previously trained members. Also in contrast to the previous approaches, boosting is able to reduce both variance *and bias* in the prediction [13]. This is due to the fact that more emphasis is put on data points which are misclassified by previously trained committee members.

The original boosting approach, *boosting by filtering,* is due to Schapire [14]. Here, three neural networks are used and the existence of an oracle is assumed which can produce an arbitrary number of training data points. The first neural network is trained on K training data points. Then the second neural network is also trained on K training data points. These training data are selected from the pool of training data (or the oracle) such that half of them are classified correctly and half of them are classified incorrectly by the first neural network. The third network is trained only on data on which network one and two disagree. The classification is then achieved by a majority vote of the three neural networks. Note that the second network obtains 50% of patterns for training which were misclassified by network one and that network three only obtains critical patterns in the sense that network one and two disagree on those patterns. The original motivation for boosting came out of PAC-learning theory (PAC stands for probably approximately correct). The theory only requires that the committee members are weak learners, i.e., that that the learners with high probability produce better results than random classification.

Boosting by filtering requires an oracle or at least a very large training data set: one needs, for example, a large training data set to obtain K patterns on which network one and two disagree. AdaBoost (adaptive boosting) is a combination of the ideas behind boosting and bagging and does not have a demand on a large training data set [15]. Many variants of AdaBoost have been suggested in the literature. Here, we present the algorithm in the form described by Breiman [7] which is an instance of *boosting by subsampling*:

Let $\{P(1), \ldots, P(K)\}$ *be a set of probabilities defined for each training pattern.*
Initialized with $P(j) = 1/K$.
FOR $i = 1, \ldots, M$

1. *Train the* i*-th member by taking bootstrap samples from the original training data set with replacement following probability distribution* $\{P(1), \ldots, P(K)\}$.

2. *Let* $d(j) = 1$ *if the* j*th pattern was classified* in*correctly and zero otherwise*

3. *Let*

   $$\epsilon_i = \sum_{j=1}^{K} d(j)p(j) \quad \textit{and} \quad \beta_i = (1 - \epsilon_i)/\epsilon_i$$

4. *The updated probabilities are*

   $$P(j) = cP(j)\beta_i^{d(j)}$$

   where c *normalizes the probabilities.*

END.

The committee output is calculated as a majority vote

$$\widehat{class}(x) = \arg\max_j \sum_{i=1}^{M} \log(\beta_i) f_{i,class=j}(x)$$

with weight $\log(\beta_i)$.

As in bagging, committee members are trained using bootstrap samples, but here, the probability of selecting a sample depends on previously trained classifiers. If a pattern j was misclassified by the previous classifier ($d(j) = 1$), it will be picked with higher probability for training the following classifier. Also, if the previous classifier performed in general well (indicated by a large β_i) the probabilities will be shifted more stringently to favor misclassified samples. Finally, the weight of the i-th classifier in the committee voting is $\log \beta_i$, putting more emphasis on well-performing committee members.

Table 1 shows results from Breiman [7]. Committee members consisted of CART decision trees. It can be seen that AdaBoost typically outperforms bagging except when a significant overlap in the classes exists, as in the diabetes data set. Here, AdaBoost tends to assign significant resources to learn to predict outliers. This is in in accordance with results reported in [10]. There it was shown that AdaBoost provides better performance in settings with little classification noise, but that bagging is superior, when class labels are ambiguous.

Another variant of AdaBoost is *boosting by reweighting*. In this deterministic version, the $\{P(1), \ldots, P(K)\}$ are not used for resampling but are used as weights for data points. In [13], several variants of this deterministic version are described. Furthermore, Adaboost can be used in context with classifiers which produce "confidence-rates" predictions (e.g. class probabilities) and can also be applied to regression [16].

Table 1: Test set error in %

Data Set	AdaBoost	Bagging
heart	1.1	2.8
breast cancer	3.2	3.7
ionosphere	6.4	7.9
diabetes	26.6	23.9
glass	22.0	23.2
soybean	5.8	6.8
letters	3.4	6.4
satellite	8.8	10.3
shuttle	.007	.014
dna	4.2	5.0
digit	6.2	10.5

Recent research shows that there is a connection between AdaBoost and large margin classifiers (i.e., the support vector machine (SVM), see the corresponding book chapter): both methods find a linear combination in a high-dimensional space which has a large margin on the instances in the sample [17, 18]. A large margin provides superior generalization for the case that classes don't overlap. For noisy data, Rätsch et al. have introduced regularized AdaBoost [19] and ν-Arc [20]. Both are boosting algorithms which tolerate outliers in classification and display improved performance for cases with overlapping classes.

It was also recently shown that AdaBoost can be seen as performing gradient descent in an error function with respect to the margin, resp. that AdaBoost can be interpreted as an approximation to additive modeling by performing gradient descent on certain cost- functionals [22, 21, 13, 23, 16].[1]

2.5 Concluding Remarks

The generalization performance for the various committee machines is certainly impressive. Committee machines improve performance when the individual members have low bias and are decorrelated. The particular feature of boosting is that it reduces both bias and variance. Due to its superior performance, boosting (in the form of AdaBoost) is currently the most frequently used algorithm of the ones described.

[1]The population version of AdaBoost of Friedman et al. [13] builds an additive logistic regression model via Newton-like updates for minimizing $E(e^{-y\hat{f}(x)})$ where E is the expectation taken over the population.

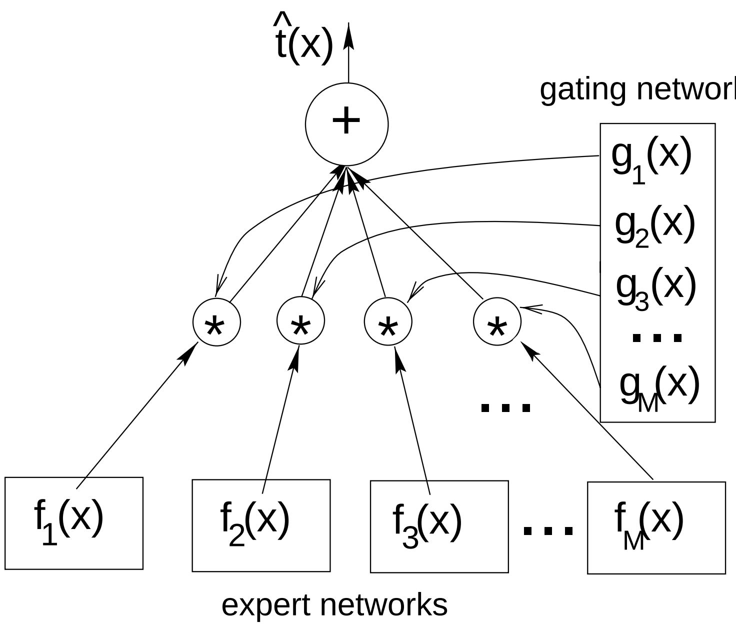


Figure 1: The ME architecture. The outputs of the gating network modulate the outputs of the expert neural networks.

Finally, we also would like to mention *stacking* which is one of the earliest approaches to committee machines. In stacking, the weights g_i are determined after training the committee members, typically by using leave-one-out cross validation. Stacking was introduced by Wolpert [24] and extended by Breiman. The general perception is that the results using stacking are somewhat mixed.

3 The Mixture of Experts and its Variants

3.1 Mixtures of Experts

The initial motivation for developing the mixture of experts (ME) approach was to design a system in which different neural networks are responsible for modeling different regions in input space. This modularity leads to greater modeling capability and potentially to a meaningful and interpretable segmentation of the map. In the ME approach, the weights become input-dependent *weighting functions* [25]

$$\hat{t}(x) = \sum_{i=1}^{M} g_i(x) f_i(x). \tag{5}$$

The committee members $f_i(x)$ are "expert" neural networks and the $g_i(x)$ —which are positive and sum to one— are generated using a gating network. The gating network determines which expert is responsible for the different regions in input space. Figure 1 illustrates how the gating network and the expert neural networks interact.

The most important training algorithm for the ME is derived using the following probabilistic assumptions. Given an input x, expert network i is selected with probability $g_i(x)$. An output is generated by adding Gaussian noise with variance $\sigma_i(x)^2$ to the output of the expert network $f_i(x)$. Since it is unknown in the data which expert network produced the output, the probability of an output given the input is a mixture distribution of the form

$$P(y|x) = \sum_{i=1}^{M} g_i(x)\; G\left(y; f_i(x), \sigma_i(x)^2\right) \quad \text{which yields} \quad E(y|x) = \sum_{i=1}^{M} g_i(x)\; f_i(x),$$

where $G\left(y; f_i(x), \sigma_i(x)^2\right)$ is the notation for a Gaussian probability density centered at $f_i(x)$, with variance $\sigma_i(x)^2$, evaluated at y. By looking at the previous equations, it becomes clear that the ME can also be considered to be a flexible model for conditional probability densities [26, 27]. Note that in this most general formulation, the outputs of the gating network, the expert neural networks and the noise variance are functions of the input. To ensure that the $g_i(x)$ are positive and sum to one and that the standard deviation is positive one parameterizes

$$g_i(x) = \frac{\exp h_i(x)}{\sum_{j=1}^{M} \exp h_j(x)} \quad \text{and} \quad \sigma_i(x) = \exp(s_i(x))$$

where the functions $h_i(x)$ and $s_i(x)$ are typically modeled as neural networks. For training the ME system, a maximum likelihood error function is used. The contribution of the j-th pattern to the log-likelihood function is

$$l_j = \log \sum_{i=1}^{M} g_i(x_j) G\left(y_j; f_i(x_j), \sigma_i(x_j)^2\right).$$

Following the derivation in [27], the gradients of l_j with respect to the outputs of the various networks become

$$\frac{\partial l_j}{\partial f_i(x_j)} = P(i|x_j, y_j) \frac{y_j - f_i(x_j)}{\sigma_i(x_j)^2}$$

$$\frac{\partial l_j}{\partial h_i(x_j)} = P(i|x_j, y_j) - g_i(x_j)$$

$$\frac{\partial l_j}{\partial s_i(x_j)} = P(i|x_j, y_j) \left(\frac{(y_j - f_i(x_j))^2}{\sigma_i(x_j)^2} - 1 \right)$$

where the probability that the i-th expert generated data point (x_j, y_j) is calculated as

$$P(i|x_j, y_j) = \frac{g_i(x_j)\; G\left(y_j; f_i(x_j), \sigma_i(x_j)^2\right)}{\sum_{i=1}^{M} g_i(x_j)\; G\left(y_j; f_i(x_j), \sigma_i(x_j)^2\right)}$$

using the current parameter estimates.

The adaptation rules simplify if both $h_i(x)$ and $f_i(x)$ are linear functions of the inputs [28]. In [29] a variant is described in which both $h_i(x)$ and $f_i(x)$ are modeled using Gaussian processes. For the $f_i(x)$, Gaussian processes with different bandwidths are used such that, input dependent, the expert with the appropriate bandwidth is selected by the gating network.

3.2 Extensions to the ME Approach

3.2.1 Hierarchical Mixtures of Experts

The hierarchical mixtures of experts (HMEs) have two (or more) layers of gating networks and their output is calculated as

$$\hat{t}(x) = \sum_{i=1}^{M} g_i(x) \sum_{j=1}^{N(i)} g_{i,j}(x) f_{i,j}(x). \tag{6}$$

where

$$g_{i,j}(x) = \frac{\exp h_{i,j}(x)}{\sum_{j=1}^{N(i)} \exp h_{i,j}(x)}.$$

In the HME, a weighted combination of the outputs of M ME networks is formed. $N(i)$ is the number of experts in the i-th ME network. In the HME approach, $h_i(x)$, $h_{i,j}(x)$ and $f_i(x)$ are typically linear functions. By using several layers of gating networks in the HME, one obtains large modeling flexibility despite the simple linear structure of the expert networks. On the other hand, by using linear models, the HME can be trained using a expectation maximization (EM) algorithm which exhibits fast convergence.

The HME can be seen as a soft decision tree and (hard) decision tree learning algorithms, such as the CART algorithm, can be used to initialize the HME. The HME was developed by Jordan and Jacobs [30]. A very extensive discussion of the HME is found in [28].

3.2.2 Alternative Training Procedures

A number of variations of the architecture and the training procedure for the ME have been reported. The learning procedure of the last section produces *competing* MEs in the sense that the committee members compete for the responsibility for the data. The *cooperating* MEs are achieved if the squared difference between output data and predictions $\sum_{j=1}^{K}(y_j - \hat{t}(x_j))^2$ is minimized. Here, the solutions are typically less modular and the model has a larger tendency to overtraining.

Another variant starts from the joint probability model

$$P(i)P(x|i)P(y|x,i)$$

where $P(i)$ is the prior probability of expert i, $P(x|i)$ is the probability density of input x for expert i and $P(y|x,i)$ is the probability density of y given x and expert i. If we assume that the latter is modeled as before, we obtain Equation 5 with

$$g_i(x) = P(i|x) = \frac{P(i)P(x|i)}{\sum_{j=1}^{M} P(j)P(x|j)}. \tag{7}$$

In some applications the experts are trained *a priori* on different data sets. In this case we can estimate $P(i)$ as the fraction of data used to train expert i, and $P(x|i)$ as the input density of the data used to train expert i. In [32] it was suggested to model the latter as a mixture of

Gaussians. In [33] and [34], approaches to generating the HME model based on joint probability models are described.

A very simple ME variant is obtained by modeling the joint input-output probability of $(x, y)'$ as a mixture of Gaussians. If we calculate the expected value of y given x, we obtain an ME network with linear experts and the gating network of Equation 7 where all conditional probability densities are Gaussians. The advantage of this simple variant is that it can be described in terms of simple probabilistic rules which can be formulated by a domain expert. Various combinations of the ME network with domain expert knowledge are described in [31, 33].

3.3 Concluding Remarks

The ME is typically not used if the only goal is prediction accuracy, but is applied in cases where its unique modeling properties, interpretability and fast learning (in the case of the HME), are of interest. The capability of the ME to model a large class of conditional probability densities is useful in the estimation of risk [27], in optimal control and portfolio management [26], and in graphical models [35]. In some applications, the ME has been shown to lead to interesting segmentations of the map, as in the cognitive modeling of motor control [36].

Of the many generalizations of the ME approach, we want to mention the work of Jacobs and Tanner [37] who generalize the ME approach to a wider class of statistical mixture models, i.e., mixtures of distributions of the exponential family, mixtures of hidden Markov models, mixtures of marginal models, mixtures of Cox models, mixtures of factor models, and mixtures of trees.

4 A Bayesian Committee Machine

Consider the case that a large training data set is available. It is not obvious that a neural network trained on such a large data set makes efficient use of all the data. It might make more sense to divide the data set into M data sets, train M neural networks on the data sets and then to combine their predictions using a committee machine (see Figure 2, left). This approach is even more appropriate if the estimators are kernel based systems such as Gaussian process regression systems, smoothing splines or support vector machines. Parameter estimation for those systems requires operation whose computational complexity grows quickly with the size of the training data set such that those systems are not well suited for large data sets. An obvious solution would be to split the training data set into M data sets, train individual estimators on the partitioned data sets and then combine their predictions. The computationally complexity of such an approach grows only linearly with the size of the training data set if M grows proportional to the size of the training data set. Important questions are: does one loose prediction performance by this procedure and what is a good combination scheme. In this section we analyze this approach from a Bayesian point of view. It turns out that the optimal combining weights can be calculated based on the covariance structure and that for the optimal prediction of the committee at a given input x, the prediction of the members at inputs *other* than x can be exploited. Note that in previous sections only predictions at x were used in the committee machines. A main result of this work is that the quality of the predictions depends critically on the number of different inputs which contribute to the prediction. If this number is identical to the effective number of parameters then the performance of the committee machine is very close to the performance of one system trained on all data.

This section is different in character than the previous sections. Whereas in previous sections, we provided an overview of the state of the art of the respective topics, this section almost exclusively presents recent results of the author [38, 39].

4.1 Theoretical Foundations

Let x be a vector of input variables and let y be the output variable. We assume that, given a function $f(x)$, the output data are (conditionally) independent with conditional density $P(y|f(x))$.

Furthermore, let $X^q = \{x_1^q, \ldots, x_{N_Q}^q\}$ denote a set of N_Q test points (superscript q for query) and let $f^q = (f_1^q, \ldots, f_{N_Q}^q)$ be the vector of the corresponding unknown response variables. Note that we query the system at a *set* of query points (or test points) at the same time which will be of importance later on.

As indicated in the introduction, we now assume a setting where instead of training one estimator using all the data, we split up the data into M data sets $D = \{D^1, \ldots, D^M\}$ (of typically approximately same size) and train M estimators on the partitioned training data sets. Let $\bar{D}^i = D \setminus D^i$ denote the data which are not in D^i.

Then we have in general[2]

$$P(f^q|\bar{D}^i, D^i) \propto P(f^q)P(\bar{D}^i|f^q)P(D^i|\bar{D}^i, f^q).$$

Now we would like to approximate

$$P(D^i|\bar{D}^i, f^q) \approx P(D^i|f^q). \tag{8}$$

This is not true in general unless f^q contains the targets at the input locations of all the training data: only then all outputs in the training data are independent. The approximation in Equation 8 becomes more accurate when

- N_Q is large — since then f^q determines f everywhere
- the correlation between the outputs in $\bar{D}^i$ and D^i is small, for example if inputs in those sets are spatially separated from each other
- when the size of the data set in D^i is large since then those data become more independent on average.

Using the approximation and applying Bayes' formula, we obtain

$$P(f^q|\mathcal{D}^{i-1}, D^i) \approx const \times \frac{P(f^q|\mathcal{D}^{i-1})P(f^q|D^i)}{P(f^q)} \tag{9}$$

such that we can achieve an approximate predictive density

$$\hat{P}(f^q|D) = const \times \frac{\prod_{i=1}^{M} P(f^q|D^i)}{P(f^q)^{M-1}} \tag{10}$$

[2]We assume that inputs are given.

where *const* is a normalizing constant. The posterior predictive probability densities are simply multiplied. Note that since we multiply posterior probability densities, we have to divide by the priors $M - 1$ times. This general formula can be applied to the combination of any suitable Bayesian estimator.

4.2 The BCM

In the case that the predictive densities $P(f^q|D^i)$ and the prior densities are Gaussian (or can be approximated reasonably well by a Gaussian), Equation 9 takes on an especially simple form. Let's assume that the *a priori* predictive density at the N_Q query points is a Gaussian with zero mean and covariance Σ^{qq} and the posterior predictive density for each committee member is a Gaussian with mean $E(f^q|D^i)$ and covariance $cov(f^q|D^i)$. In this case we achieve [38] ($\hat{E}$ and $\widehat{cov}$ are calculated with respect to the approximate density $\hat{P}$)

$$\hat{E}(f^q|D) = \frac{1}{C} \sum_{i=1}^{M} cov(f^q|D^i)^{-1} E(f^q|D^i) \tag{11}$$

with

$$C = \widehat{cov}(f^q|D)^{-1} = -(M-1)(\Sigma^{qq})^{-1} + \sum_{i=1}^{M} cov(f^q|D^i)^{-1}. \tag{12}$$

We recognize that the predictions of each committee member i are weighted by the inverse covariance of its prediction. But note that we do not compute the covariance between the committee members but the covariance at the N_Q query points! This means that predictive densities at all query points contribute to the prediction of the BCM at a given query point (see Figure 2, right). This way of combining the predictions of the committee members is named the Bayesian committee machine (BCM). An intuitive appealing effect is that by weighting the predictions of the committee members by the inverse covariance, committee members which are uncertain about their predictions are automatically weighted less than committee members which are certain about their predictions.

4.3 Experiments

In the experiments, we applied the BCM to Gaussian process regression. Gaussian process regression is particularly suitable since prior and posterior densities are Gaussian distributed and the covariance matrices which are required for the BCM are easily calculated. A short introduction into Gaussian process regression can be found in the Appendix.

Figure 3 (right) shows results of applying the BCM to Gaussian process regression for a large artificial data set [38]. Note that the BCM with $K = 60000$ training data points achieves results unobtainable with simple Gaussian process regression which is limited to a training data set of approximately $K = 1000$.

4.4 Concluding Remarks

The previous discussion on the BCM has focussed on regression. In the form of the generalized BCM (GBCM), the approach has been extended towards classification, the prediction of counts,

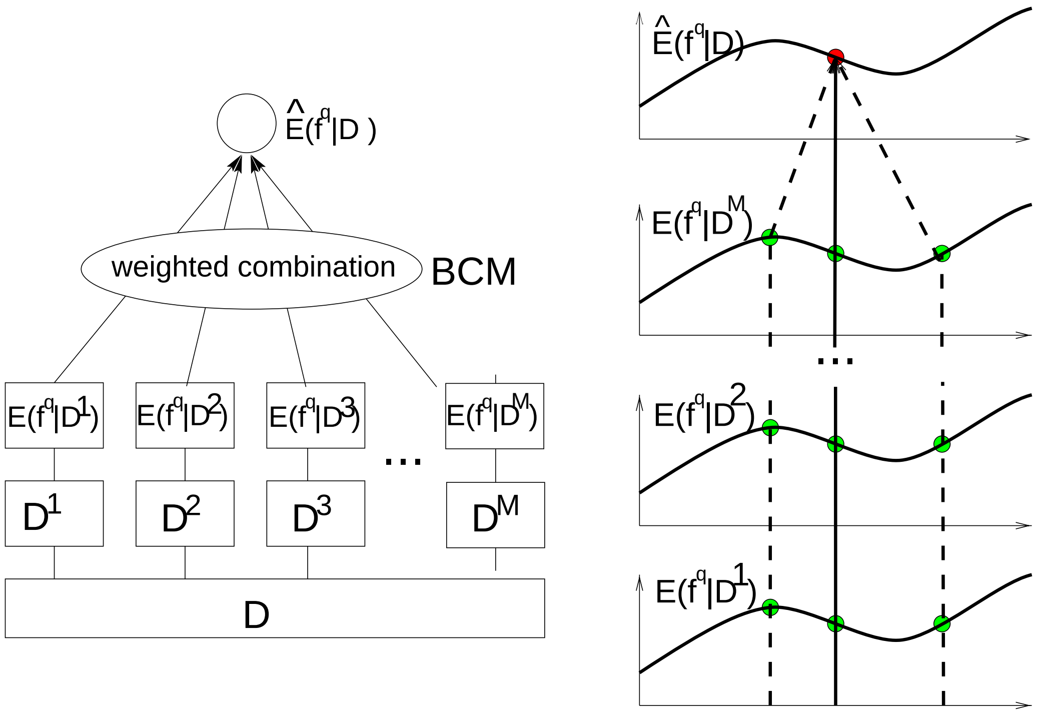
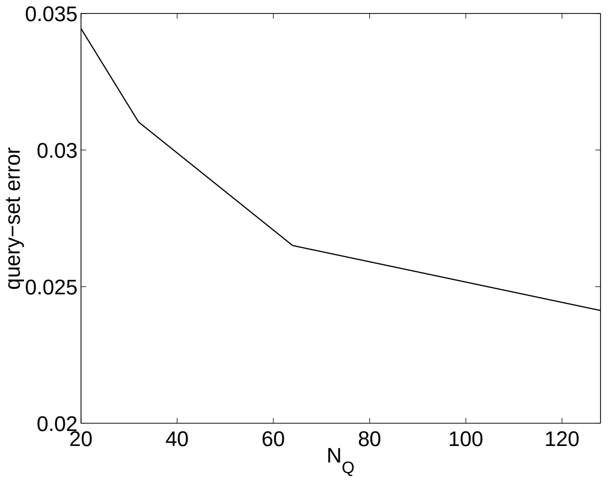


Figure 2: Left: In the BCM, data sets are partitioned and the committee members estimate the targets based on the partitioned data sets. The prediction of the committee is formed by a weighted combination of the predictions. Right: In most committee machines, the prediction at a given input is calculated by a combination of the predictions of the committee members at the same input (vertical continuous line). In the BCM, predictions at other inputs are also used (vertical dashed lines).



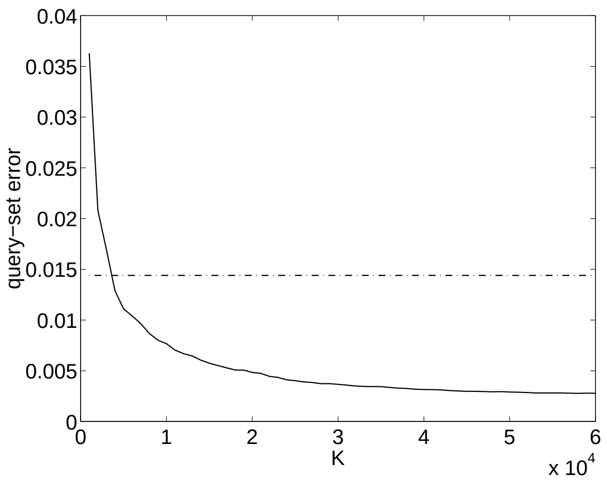


Figure 3: Left: The mean squared query-set error as a function of the number of query points N_Q with $K = 1000$ training data points and for $K/M = 100$ data points in the training data set of each committee member. Note that for small N_Q, the performance deteriorates.
Right: The test of the BCM algorithm using an artificial data set. The continuous line is the mean squared query-set error for $N_Q = 1000$ and $K/M = 1000$ as a function of the size of the training data set K. The dash dotted line shows the error of Gaussian process regression (M=1) with $K = 1000$, approximately the largest training data size suitable for Gaussian process regression. Note the great improvement with the BCM which can use the larger data sets.

the prediction of lifetimes, and other applications which can be derived from the exponential family of distributions [39]. The support vector machine is closely related to Gaussian processes and can also be used in the BCM [40]. Furthermore, it is possible to derive online Kalman filter versions of the BCM which only requires one path through the data set and only requires the storage of a matrix of the dimension of the number of query or test points [38]. After training, the prediction at additional test points only requires resources dependent on the number of query points but is independent of the size of the training data set. An interesting question is how many query points ideally are required. It turns out that the number of query points should be equal to or larger than the effective number of parameters of the map. If this is the case the query points uniquely define the map (if appropriately chosen) and approximation in Equation 8 becomes an equality [38] (see Figure 3, left).

Since the BCM has the form of a committee machine, it might be possible to further improve performance by connecting it with ideas from Section 2. The issue of boosting the BCM is a focus of current work.

Finally, it is also possible to derive the BCM solution from a non-Bayesian perspective where the covariance matrices are derived from variations over repeated experiments (i.e., variations over data sets). The goal is to form a linear combination of the predictions at the query points

$$\hat{f}_i^q = \sum_{k=1}^{M} \sum_{j=1}^{N_Q} g_{i,k}(x_j) f_k^q(x_j)$$

such that the expected error is minimum. Here, $\hat{f}_i^q$ is the estimate of the committee for input x_i, $g_{i,k}(x_j)$ is the weight of query point j of committee member k to predict query point i and $f_k^q(x_j)$ is the prediction of committee member k for input x_j. We use the assumption that the individual committee members are unbiased and enforce the constraint that the combined system is unbiased by requiring that

$$\sum_{k=1}^{M} g_{i,k}(x_i) = 1 \quad \text{and} \quad \sum_{k=1}^{M} g_{i,k}(x_j) = 0 \quad \forall\, j \neq i.$$

As solution we obtain the BCM with $(\Sigma^{qq})^{-1} \rightarrow 0$ (negligible prior). Details can be found in [38].

5 Conclusions

The research on committee machines is still very young but has already produced a number of interesting architectures and algorithms. Due to the limited space, we naturally could not provide a comprehensive overview covering all aspects of committee machines. As examples, boosting has been analyzed from the perspectives of PAC-learning, game theory and "conventional" statistics. Furthermore, committee machines have been applied to density estimation [41, 42], hidden Markov models and a variety of statistical mixture models. Also, we completely left out approaches to committee machines originating from statistical physics and modular approaches for sensory fusion and control. Finally, there are interesting common aspects between committee machines and biological systems: Committee machines, in general, are very tolerant versus the breakdown of any of its components and, similarly to biological systems, system performance only deteriorates dramatically if a large number of committee members (as neural modules) are malfunctioning.

In their many facets, committee machines have made important contributions to various branches of neural computation, machine learning, and statistics and are becoming increasingly popular in the KDD-community. They have opened up a new dimension in machine learning since committees can be formed using virtually any prediction system. As demonstrated by the numerous contributions from various communities, committee machines can be understood and analyzed from a large number of different view points, often revealing new interesting insights. We can expect numerous novel architectures, algorithms, applications and theoretical insights for the future.

Acknowledgements

Valuable comments by Salvatore Ingrassia and Michal Skubacz are greatfully acknowledged.

6 Appendix: Gaussian Process Regression

In contrast to the usual parameterized approach to regression, in Gaussian process regression we specify the prior model directly in function space. In particular, we assume that *a priori* f is Gaussian distributed (in fact it would be an infinite-dimensional Gaussian) with zero mean and a covariance $cov(f(x_1), f(x_2)) = \sigma_{x_1,x_2}$. We assume that we can only measure a noisy version of f

$$y(x) = f(x) + \epsilon(x)$$

where $\epsilon(x)$ is independent Gaussian distributed noise with zero mean and variance $\sigma_\psi^2(x)$.

Let $(\Sigma^{mm})_{ij} = \sigma_{x_i^m,x_j^m}$ be the covariance matrix of the measurements, $\Psi^{mm} = \sigma_\psi^2(x)I$ be the noise variance between the outputs, $(\Sigma^{qq})_{ij} = \sigma_{x_i^q,x_j^q}$ be the covariance matrix of the query points and $(\Sigma^{qm})_{ij} = \sigma_{x_i^q,x_j^m}$ be the covariance matrix between training data and query data. I is the K-dimensional unit matrix.

Under these assumptions, the conditional density of the response variables at the query points is Gaussian distributed with mean

$$E(f^q|D) = \Sigma^{qm}(\Psi^{mm} + \Sigma^{mm})^{-1}y^m, \tag{13}$$

and covariance

$$cov(f^q|D) = \Sigma^{qq} - \Sigma^{qm}\ (\Psi^{mm} + \Sigma^{mm})^{-1}\ (\Sigma^{qm})'. \tag{14}$$

Note that for the $i-$th query point we obtain

$$E(f_i^q|D) = \sum_{j=1}^{K} \sigma_{x_i^q,x_j^m}\ v_j \tag{15}$$

where v_j is the $j-$th component of the vector $(\Psi^{mm} + \Sigma^{mm})^{-1}y^m$. The last equation describes the weighted superposition of kernel functions $b_j(x_i^q) = \sigma_{x_i^q,x_j^m}$ which are defined for each training data point and is equivalent to some solutions obtained for kriging, regularization neural networks and smoothing splines. The experimenter has to specify the positive definite covariance matrix. A common choice is that $\sigma_{x_i,x_j} = \exp(-1/(2\gamma^2)||x_i - x_j||^2)$ with $\gamma > 0$ such that we obtain Gaussian basis functions, although other positive definite covariance functions are also used.

References

[1] Sharkey, A. J. C., *Combining artificial neural nets,* Springer, 1999.

[2] Granger C. W. J., Combining forecasts-twenty years later, *Journal of forecasting,* 8, 167, 1989.

[3] Bernardo, J. M., and Smith, A. F. M., *Bayesian theory,* New York: J. Wiley & Sons, 1993.

[4] Quinlan, J. R., *Thirteenth National Conference on Artificial Intelligence (AAAI-96)* AAAI press, 1996.

[5] Meir, R., Bias, Variance and the Combination of Least Squares Estimators. in *Advances in Neural Information Processing Systems 7,* Tesauro, G., Touretzky, D. S., and Leen T. K., Eds., MIT Press, 295, 1995.

[6] Taniguchi, M. and Tresp, V., Averaging regularized estimators, *Neural Computation,* 9, 1163, 1997.

[7] Breiman, L., Combining predictors, in *Combining artificial neural nets,* Sharkey, A. J. C., Ed., Springer, 31, 1999.

[8] Perrone, M. P., *Improving regression estimates: averaging methods for variance reduction with extensions to general convex measure optimization,* PhD thesis, Brown University, 1993.

[9] Krogh, A. and Vedelsby, J., Neural network ensembles, cross validation, and active learning, in *Advances in Neural Information Processing Systems 7,* Tesauro, G., Touretzky, D. S., and Leen T. K., Eds., MIT Press, 231, 1995.

[10] Dietterich, T. G., An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization, *Machine Learning,* 40, 130, 2000.

[11] Breiman, L., Randomizing outputs to increase prediction accuracy, *Machine Learning,* 40, 229, 2000.

[12] Raviv, Y., and Intrator, N., Variance reduction via noise and bias constraints, in *Combining artificial neural nets,* Sharkey, A. J. C., Ed., Springer, 31, 1999.

[13] Friedman J., Hastie T., and Tibshirani, R., Additive logistic regression: a statistical view of boosting, *Annals of Statistics,* to appear.

[14] Schapire R. E., The strength of weak learnability, *Machine Learning,* 5, 197, 1990.

[15] Freund Y. and Schapire R. E., Experiments with a new boosting algorithm, in *Machine Learning: Proceedings of the Thirteenth International Conference,* 148, 1996.

[16] Zemel, R. S. and Pitassi, T., A gradient-based boosting algorithm for regression problems, in *Advances in Neural Information Processing Systems 13,* Leen, T. K., Dietterich, T. G., and Tresp V., Eds., MIT press, 2001.

[17] Schapire R. E., Freund Y., Bartlett P., and Lee, W. S. Boosting the margin: A new explanation for the effectiveness of voting methods, *The Annals of Statistics,* 1651, 1998.

[18] Drucker, H., Boosting neural networks, in *Combining artificial neural nets,* Sharkey, A. J. C., Ed., Springer, 31, 1999.

[19] Rätsch, G., Onoda, T., and Müller, K.-R., Regularizing AdaBoost, in *Advances in Neural Information Processing Systems 11,* Kearns, M. S., Solla, S. A., and Cohn, D. A. Eds., MIT Press, 564, 1999.

[20] Rätsch, G., Schölkopf, B., Smola, A., Müller, K.-R., Onoda, T., and Mika, A., ν-Arc: ensemble learning in the presence of noise, in *Advances in Neural Information Processing Systems 12,* Solla, S. A., Leen T. K., and M "uller, K. R. Eds., MIT Press, 561, 2000.

[21] Schapire, R. E. and Singer, Y., Improved boosting algorithms using confidence-rated predictions, in *Proceedings of the eleventh annual conference on computational learning theory,* 1998.

[22] Breiman, L., Predicting games and arcing algorithms, TR. 504, Statistics Department, University of California, December, 1997.

[23] Mason, L., Baxter, J., Bartlett, P., and Frean, M., Boosting algorithms as gradient descent, in *Advances in Neural Information Processing Systems 12,* Solla, S. A., Leen T. K., and Müller, K. R. Eds., MIT Press, 512, 2000.

[24] Wolpert, D. H., Stacked generalization, *Neural Networks*, 5, 241, 1992.

[25] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, J. E., Adaptive mixtures of local experts, *Neural Computation,* 3, 79, 1991

[26] Neuneier, R. F., Hergert, W., Finnof, W., and Ormoneit, D., Estimation of conditional densities: a comparison of approaches, *Proceedings of ICANN*94,* Springer, 1, 689, 1994.

[27] Bishop, C. M., *Neural neural networks for pattern recognition,* Oxford: Clarendon Press, 1995.

[28] Haykin, S., *Neural Networks, a comprehensive foundation,* 2nd edition, Prentice Hall, 1999.

[29] Tresp, V., Mixtures of Gaussian processes, in *Advances in Neural Information Processing Systems 13,* Leen, T. K., Dietterich, T. G., and Tresp V., Eds., MIT press, 2001.

[30] Jordan, M. I., and Jacobs, R. A., *Hierarchical mixtures of experts and the EM algorithm, Neural Computation,* 6, 181, 1994.

[31] Tresp, V., Hollatz, J., and Ahmad, S., Network structuring and training using rule-based knowledge, in *Advances in Neural Information Processing Systems 5,* Giles, C. L., Hanson S. J., and Cowan J. D., Eds., Morgan Kaufman, 871, 1993.

[32] Tresp, V., and Taniguchi, M., Combining estimators using non-constant weighting functions, in *Advances in Neural Information Processing Systems 7,* Tesauro, G., Touretzky, D. S., and Leen T. K., Eds., MIT Press, 419, 1995.

[33] Tresp, V., Hollatz, J., and Ahmad, S., Representing Probabilistic Rules with Networks of Gaussian Basis Functions, *Machine Learning,* 27, 173, 1997.

[34] Xu, L., Jordan, M. I., and Hinton, G. E., An alternative model for mixtures of experts, in *Advances in Neural Information Processing Systems 7,* Tesauro, G., Touretzky, D. S., and Leen T. K., Eds., MIT Press, 633, 1995.

[35] Hofmann, R. and Tresp, V., Discovering structure in continuous variables using Bayesian networks, in *Advances in Neural Information Processing Systems 8,* Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., Eds., MIT Press, 500, 1996.

[36] Brashers-Krug, T., Shadmehr, R., and Todorov, E., M., and Jacobs, R., Catastrophic inference in human motor learning, in *Advances in Neural Information Processing Systems 7,* Tesauro, G., Touretzky, D. S., and Leen T. K., Eds., MIT Press, 19, 1995.

[37] Jacobs, R. and Tanner, M., Mixtures of X, in *Combining artificial neural nets,* Sharkey, A. J. C., Ed., Springer, 31, 1999.

[38] Tresp, V., The Bayesian committee machine, *Neural Computation,* 12, 2000.

[39] Tresp, V., The generalized Bayesian committee machine, *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD-2000,* 130, 2000.

[40] Schwaighofer, A., and Tresp, V., The Bayesian committee support vector machine, manuscript, submitted for publication, 2001.

[41] Ormoneit, D. and Tresp, V., Averaging, maximum penalized likelihood and Bayesian estimation for improving Gaussian mixture probability density estimates, *IEEE Transactions on Neural Networks,* 9, 639, 1998.

[42] Smyth, P., and Wolpert, D., Stacked density estimation, in *Advances in Neural Information Processing Systems 10,* Jordan, M. I., Kearns, M. J., and Solla A. S., Eds., MIT Press, 668, 1998.