LETTER Communicated by Robert Jacobs

Boosted Mixture of Experts: An Ensemble Learning Scheme

Ran Avnimelech
Nathan Intrator
Department of Computer Science, Sackler Faculty of Exact Sciences, Tel-Aviv University, Tel-Aviv, Israel

We present a new supervised learning procedure for ensemble machines, in which outputs of predictors, trained on different distributions, are combined by a dynamic classifier combination model. This procedure may be viewed as either a version of mixture of experts (Jacobs, Jordan, Nowlan, & Hinton, 1991), applied to classification, or a variant of the boosting algorithm (Schapire, 1990). As a variant of the mixture of experts, it can be made appropriate for general classification and regression problems by initializing the partition of the data set to different experts in a boostlike manner. If viewed as a variant of the boosting algorithm, its main gain is the use of a dynamic combination model for the outputs of the networks. Results are demonstrated on a synthetic example and a digit recognition task from the NIST database and compared with classical ensemble approaches.

1 Introduction

The mixture-of-experts approach has great potential for improving performance in machine learning. The improved classification and regression performance achieved by using an ensemble of networks rather than a single net for classification and regression tasks is well established (Hansen & Salamon, 1990; Wolpert, 1992; Breiman, 1996c; Perrone & Cooper, 1993; Raviv & Intrator, 1996). Earlier work focused on voting schemes—majority and plurality—but in later studies, averaging of the outputs was usually found to be superior.

Advanced methods for combining the output of different classifiers are suggested in Ho, Hull, and Srihari (1994). Logistic regression (perceptron) is applied on the output of the classifiers to achieve better results than simple averaging; furthermore, the static combination of experts is replaced by a dynamic model (DCS), so that only one of several logistic regression functions is chosen, according to the input or to the classifier outputs. Generally there are two approaches to combining outputs of different classifiers: selection, or choosing the locally best classifier, and averaging, or reducing the variance by combining outputs that are not fully correlated. DCS and other methods combine these approaches by using a dynamic weighted average.

Stacking is another framework for combining estimators that uses a nonsymmetric combination (Wolpert, 1992; Breiman, 1996c). The principle is to use several levels of learners, in a manner that is basically an extension of choosing a learner by cross-validation. To avoid training the combination level on overfit outputs of the lower-level learners, each input pattern to the combination learner is extracted by copies of the learners trained on the data, excluding that pattern. The algorithm is applicable for either multiple learners or a single learner. The popular form of stacking uses two levels with a linear combination model, possibly with constrained coefficients (e.g., nonnegative, sum to 1).

Other methods use dynamic linear combination models, using a confidence measure of the ensemble members regarding each pattern. Different measures of the confidence of each predictor can be used for determining the relative contribution of each expert (Tresp & Taniguchi, 1995; Shimshoni & Intrator, 1996).

All of these algorithms train the individual classifiers independently for the same goal. More specifically, the different parts of the training set that are used to train individual classifiers are all drawn from the same distribution. This holds when different types of classifiers are used, in cross-validation (Meir, 1995; Krogh & Vedelsby, 1995), or when different noisy bootstrap copies are used (Raviv & Intrator, 1996). A different approach is training the classifiers on different parts of the training set, partitioned in a manner such that their distributions differ. Such an approach, which is presented here, combines two algorithms: boosting and mixture of experts.

Sections 2 and 3 describe the boosting and adaptive mixture-of-experts algorithms. These algorithms are compared in section 4, and various ways to combine them are suggested in section 5. Following this discussion we present in section 6 the basic and advanced versions of the new algorithm. The empirical evaluation of the algorithm on a demonstration problem and on a character recognition task from the NIST database is reported in section 7.

2 Theory of Boosting

The boosting algorithm can improve the performance of learning machines (Schapire, 1990). Its theoretic basis relies on a proof of the equivalence of the strong and weak PAC (probably approximately correct) learning models. In the standard PAC model, for any distribution of patterns and for arbitrary small δ and ϵ, the learner must be able to produce a hypothesis about the underlying concept, with an error rate of at most ϵ with a probability of at least $(1-\delta)$. The weak PAC model, however, requires just $\epsilon < 1/2$—slightly better than a random guess on this two-class model.

Schapire proved the equivalence of the two models by proposing a technique for converting any weak learning algorithm (on any given distribution) to a strong learning algorithm. He termed this provably correct tech-

nique *boosting*. The basis of the technique is creating different distributions on which different subhypotheses are trained. Schapire has proved that if three such weak subhypotheses, which have an error rate of $\alpha < 1/2$ (on the respective distributions), are combined, the resulting ensemble hypothesis will have an error rate of $3\alpha^2 - 2\alpha^3$, which is smaller than α. Schapire suggested hierarchical combinations of classifiers, such that an arbitrarily low error rate can be achieved.

A procedure for creating appropriate distributions is the following: A classifier is trained on the original distribution. Fifty percent of the training set for the second classifier are patterns misclassified by the first classifier, and 50% are patterns correctly classified by it (no change in the internal distribution of each of these two groups). The third classifier is designed to break ties. Its training set contains only patterns on which the first two classifiers disagree.

Real-world machine learning tasks do not necessarily match the weak PAC model, and even if they did, the assured performance for worst-case scenario would not necessarily be higher than the practically achieved performance of simple classifiers. Still, boosting proved to be not just a theoretical technique, but also a practical tool for enhancing performance. Drucker, Schapire, and Simard (1993) demonstrated its advantage over a combination of independently trained classifiers (parallel machine) on a handwritten recognition task. Recently, boosting achieved an extremely low error rate on the same problem (Bottou et al., 1994).

Various improvements have been made to the original boosting algorithm. Freund (1990) suggested using a simpler structure for combining many subhypotheses: instead of having a tree of majority gates, all subhypotheses are presented to one majority gate. AdaBoost (Freund & Schapire, 1995) is a more advanced algorithm, in which each pattern is assigned a different probability to appear in the training set presented to the new learner. This version also prefers a flat structure for combining the classifiers rather than a hierarchical one. Another idea mentioned within the AdaBoost framework is the use of a weighted combination of the individual classifiers. Recently several applications of AdaBoost have been reported (Breiman, 1996b; Schwenk & Bengio, 1997). Breiman regards boosting as one example of an algorithm performing adaptive resampling of the training set and suggests other such algorithms. He applied these algorithms to decision trees (CARTs) on various data. Schwenk and Bengio applied Adaboost to multilayer perceptrons (MLPs) and autoencoder-based classifiers ("diabolo networks") on character recognition tasks.

3 The Mixture-of-Experts Learning Procedure

The adaptive mixture of local experts (Jacobs et al., 1991) is a learning procedure that achieves improved performance in certain problems by assigning different subtasks to different learners. Its basic idea is concurrently to train

several expert classifiers (or regression estimators) and a gating function. The gating function assigns probability to each of the experts based on the current input. In the training stage, this value states the probability of a pattern's appearing in an expert's training set. In the test stage, it defines the relative contribution of each expert to the ensemble. The training attempts to achieve two goals: (1) for a given expert, find the optimal gating function, and (2) for a given gating function, train each expert to achieve maximal performance on the distribution assigned to it by the gating function. This decomposition of the learning task motivates an expectation-maximization version of the algorithm, though simultaneous training was also used.

Much emphasis is given in this framework to making the experts local, which is a key to improving performance over ensembles of networks trained on similar distributions. A basic level of locality is achieved by targeting each expert for maximal performance on its distribution instead of having it compensate for errors of other experts. Further localization is achieved by giving higher learning rates to the better-performing expert on each pattern.

This idea was later extended into a tree structure termed hierarchical mixture of experts (HME), in which experts may be built from lower-level experts and gating functions (Jordan & Jacobs, 1992). In later work, the EM algorithm was used for training the HME (Jordan & Jacobs, 1994). Waterhouse and Robinson (1996) describe how to grow these recursive learning machines gradually. The mixture-of-experts procedure achieves superior generalization and fast learning when the learning task corresponds to different subtasks for distinct portions of the input space.

The mixture-of-experts algorithm differs from other ensemble algorithms in the relation between the combination model and the basic learners (and our algorithm follows it). Most ensemble learning algorithms, such as stacking, first train the basic predictors (or use existing predictors) and then try to tune the combination model. The mixture-of-experts algorithm trains the combination model simultaneously with the basic learners, and the current model determines the data sets provided to each learner for its further training.

4 Comparison of the Two Algorithms

Boosting and mixture of experts were developed for different types of problems and thus have different advantages and weaknesses. Any attempt to combine principles from both should address their limitations and overcome them by combining elements of the other method.

The mixture of experts is suitable when the patterns can be naturally divided into simpler (homogeneous) subsets, and the learning task in each of these subsets is not as difficult as the original one. However, real-world problems may not exhibit this property, and, furthermore, even when such a partition exists, the required gating function may be complex and the initial

stage of localizing the experts has a chicken-and-egg nature. In boosting, the distributions are selected to encourage each classifier to become an expert on patterns on which the previous classifiers err or disagree[1]—difficult patterns—while maintaining a reasonably good performance on easier patterns.

The two main advantages of the mixture of experts are localization of the different experts and use of a dynamic model for combining the outputs. In boosting, the first classifier is trained on all patterns, and the localization criterion for the distributions presented to the two other classifiers is the level of difficulty of the patterns as measured by classification performance. The limitation of this criterion is that it cannot be applied to unlabeled data, therefore disabling the use of a dynamic model based on a similar criterion.

5 Combining Boosting and HME Algorithms

There are several approaches for combining features of boosting and mixture of experts:

- *Improved boosting*. Adding a dynamic model for combining the outputs of the classifiers. (This feature is not unique to mixture of experts.)
- *Initialized mixture of experts*. The main boosting feature one would like to introduce to the mixture-of-experts framework is the ability to initialize a split of the training set to different experts.
- *Multilevel approach*. Using a mixture-of-experts classifier as the second or third boosting classifier can solve two problems: The difficult patterns may be more easily partitioned to subgroups, while the second and third boosting classifiers usually handle a more difficult problem from the original one. This approach incorporates classifier selection and classifier combination.

Waterhouse and Cook (1997) have attempted to combine boosting with the mixture of experts using the first two approaches. They report that using a dynamic model for combining boost-trained networks achieved improved performance versus simple addition. They also report that the mixture of experts was best when bootstrapped from boosted networks (bootstrapping from simple ensemble was also superior to starting from random weights).

6 The Boosted Mixture of Experts

The work presented here attempts to design a new algorithm that applies principles of both boosting and the mixture of experts and has high performance on classification or regression problems. The proposed boosted-mixture-of-experts (BME) algorithm may be considered either as a boost-

[1] More precisely, patterns on which the output may have maximal influence on the ensemble's classification.

wise initialized mixture of experts or as a variant of boosting that uses a dynamic model for combining output of the classifiers.

The main boosting feature we want to include in our scheme is the ability to initialize a split of the training set to different experts. This split is based on a difficulty criterion. In boosting, this difficulty criterion is the errors of the first classifier or the disagreement between the first two classifiers. We prefer using a confidence measure rather than errors as our difficulty criterion. This has several advantages: the size of the difficult set is more flexible (a flexible error-oriented criterion is actually error plus confidence), it focuses on the patterns that could be classified correctly, and it avoids focusing on mislabels. This also enables using other confidence-oriented methods. (Such an approach is actually used for constructing the training set of the third classifier in boosting.)

Our method includes an important component that boosting lacks: a dynamic model for combining the outputs of the classifiers. This requires a method for assigning each of the unlabeled patterns to the best-fitting classifier (or weighted combination). We follow the mixture-of-experts scheme and use the same gating function used for partitioning the data between the experts during training as the gating function for combining the outputs. Instead of training a separate gating function, we use a confidence measure, which is available for unlabeled patterns too.

6.1 The Basic Algorithm. The algorithm is designed for an arbitrary number of experts as the ensemble is constructed gradually by adding a new expert and repartitioning the data. The experts used in our work are neural nets, though any classifier with a good confidence measure is appropriate. The confidence measure is a key to achieving improved performance, and the flexibility in choosing it extends the range of applications of the algorithm.

Basically, the algorithm trains several learners on different (possibly overlapping) portions of the data. The confidence measure—$C^i(x) = C(\mathbf{o}^i(x))$—is a scalar function of the basic learner's output vector, which is used as a gating function. It determines the probability of patterns to be assigned to the data set of any learner; thus, these training sets may change as the learners evolve and their output vectors change. In addition to the confidence, the gating may be influenced from the basic reliability of each learner: $g_i(x) = w_i \cdot C^i(x)$. The reliability may be calculated by finding the optimal weighted average of the (output*confidence) of each classifier, and its value changes as the learners evolve. The output of this gating function is also used in the dynamic combination model as the coefficient assigned to each predictor for this pattern.

The confidence measure may be based on specifics of the predictor used. For an MLP performing classification, with continuous-valued output, it may be some function of the output vector. The confidence should increase as the highest output is higher and decrease as any of the other outputs

is higher. Other confidence measures, reported in machine learning literature, may also be used. Tresp and Taniguchi (1995) use various confidence measures of different predictors, in their combination model. One they use is the variance of the predictor as it is measured by the local sensitivity to changes in weights. Another approach they mention is assuming that the different predictors were trained on different data sets (e.g., American versus European digit data), and a hidden input indicates the set to which a pattern belongs. Estimating that value may be used to extract confidence information. Tresp and Taniguchi also suggest a unified approach, of which these two methods are extreme cases. Shimshoni and Intrator (1996) used base-level ensembles of several similar estimators as the different experts. The variance within each base-level ensemble indicates its confidence. A monotone function can be applied to the confidence measure to determine whether a soft or a hard partition of the data is to be used.

The confidence measure we used on a multiclass classification task was based on the difference between the two highest outputs. This is the network's estimate of its confidence margin and is a "natural" confidence measure provided by the MLP. We found that in order to encourage good localization, it was better to apply some power higher than 1 to the basic confidence measure. With a continuous-valued output, with rankings {R}, the confidence of the ith expert is $C^i(x) = [O^i_{R_1}(x) - O^i_{R_2}(x)]^n$.

The algorithm for constructing a BME consists of several procedures:

- A procedure for training a single classifier on a given training set (we used a variant of BP).
- A procedure for adding a classifier to an existing ensemble—assigning a training set for its initial training. We took a predefined portion from the training set of each of the experts, consisting of the patterns on which it was less confident.
- A refining procedure. Repartition the data according to the current confidence level of each expert on each pattern. This can be done deterministically, by assigning each pattern to the most confident expert, or stochastically, by which the probability of assigning a pattern to a certain expert is proportional to its confidence (we used the stochastic version).

The following algorithm describes how these different components fit into the constructive algorithm for creating a BME:

Algorithm.

1. *Train the first expert on all the training set.*
2. *Assign the patterns on which the current experts are not confident to the initial training set of the new expert and train it.*
3. *Refining stage:* `for i=1:N`

- *Partition the data according to the confidence of each expert on each pattern.*
- *Train each expert on its training set.*

4. If more experts are required, return to step 2.

Once the experts are trained, they may be used as an ensemble. The classifier combination model is based on the same gating function used for the localization of the experts. The exact choice of the gating function—both the confidence measure and the applied function—defines a specific variant of this algorithm. This gives the algorithm its flexibility and enables further improvement by handcrafting a confidence measure matching the specific problem (although we did not find this extra tuning necessary). The flexible nature of this algorithm makes it appropriate for most pattern recognition problems. The choice of the function may depend on specific features of the problem and of the basic learners.

The effective number of parameters used by a BME ensemble is greater than that used by an ensemble that averages similar classifiers, trained on the same data set (parallel machine). A parallel machine with k classifiers, each with $\mathcal{N}$ effective parameters, also has $\mathcal{N}$ effective parameters. A BME effectively has more parameters because of the difference between the data sets (because the confidence measure is a constant simple function of the output vector, it adds no parameters). The upper bound is $k\mathcal{N}$ parameters, but the actual number is much closer to the lower bound.

6.2 Multilevel Ensembles: Model Selection Plus Averaging. We emphasized the different advantages of two basic combination schemes: classifier selection and averaging. We argue that by applying two levels of ensembles—one for selection and the other for averaging—the advantages of each ensemble approach may be exploited in a better way than by a compromise.

Most studies state that from a certain number of classifiers, the performance of an ensemble becomes steady. When the training set is partitioned between different experts, the effect of overfit may cause a decline in the performance as the number of experts increases and the training set size for each expert becomes too small. We suggest two ways of combining ensemble averaging and expert selection to improve performance.

The first approach is training several sets of BMEs and using them in a multilevel ensemble: The output of this ensemble is the simple average of the outputs of the various BMEs, each extracted as previously described. Some of the gain here is due to overcoming the "stitch" effect: patterns in the boundaries between regions covered by different experts may yield poor performance. Using different sets of BMEs with different partitions might help overcome this.

The ability to gain from such a multilevel approach relies on the lower-

level ensemble's being a selection ensemble. For ensembles based on averaging learners trained with similar data, this would just be a larger ensemble. At the other extent, decision trees may be considered as selection-style ensembles of simpler tree predictors. Ensembles, combining the output of trees trained on bootstrapped copies of the same data (*bagging*), effectively improve performance (Breiman, 1996a). Ensemble methods that encourage diverse training sets may gain from such a method if the data partitions vary. Using a dynamic combination models makes the ensemble even more a selection-style ensemble. Therefore, this approach is most appropriate for use with the BME algorithm.

Another approach follows ideas from the query-by-committee framework (Seung, Opper, Sompolinsky, 1992; Freund, Seung, Shamir, & Tishby, 1993). According to this approach, a disagreement in an ensemble marks interesting patterns that are located in information gaps. Committees may be used as the basic experts, with the average as the expert's output and the disagreement between the committee members as a measure to the expert's confidence. It is likely that the agreement between the different members of a committee is higher because the presented patterns are more similar to those in the committee's training set. This also follows the principle used in Perrone and Cooper (1993). They suggest that in order to achieve an ensemble with minimum variance, the coefficient for each member should be inversely proportional to its variance (versus the ground truth). We assume that because of the different training sets, the members of each committee have different variances that vary in different regions of the input space. This follows the use of the internal variance in each committee as an estimate to its error rate (Shimshoni & Intrator, 1996).

7 Results

7.1 Synthetic Example. We first demonstrate the capabilities of the algorithm on a synthetic two-class two-dimensional problem (see Figure 1), to provide more intuition about the way it works. Each class is a mixture of gaussians. Patterns of the first class are drawn with a probability of 80% from the leftmost gaussian ($x \sim N(-6, 1), y \sim N(0, 1.5)$) and with probability of 20% from the lower central gaussian ($x \sim N(1, 1), y \sim N(-0.4, 0.1)$). Patterns of the second class are similarly drawn from the gaussians centered at (6,0) and $(-1, 0.4)$.

We performed tests with 2000 points drawn with equal probability from both classes. We used a simple perceptron as our basic learner. A single learner achieved a 16% error rate (all induced by the small gaussians). An ensemble composed of two to four independent learners combined by a weighted average achieved similar performance. A multilayer perceptron with two hidden units also had 16% error.

The BME ensemble used the absolute value of the perceptron output (which was in $[-1, 1]$) as its confidence score and a gating function, com-

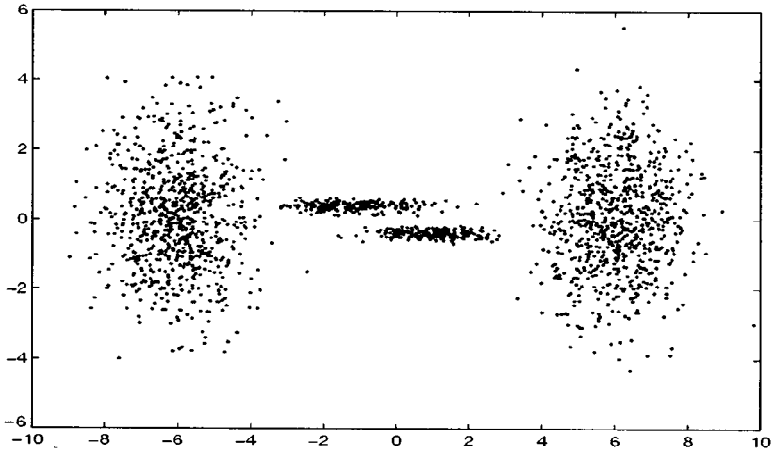

Figure 1: Input distribution of the synthetic task.

bining the confidence function and a constant coefficient for each of the two basic learners (a hard partition was used for training). The BME ensemble achieved a 3% error rate on this task. The "first" learner performs a horizontal separation: the main gaussians are classified correctly, with high confidence, and patterns in the small gaussians get a low confidence score. The second learner performs a vertical separation, but it tends to overestimate its confidence. However, the first learner is assigned a higher reliability coefficient; thus, the output of the second learner has influence only when the first one is not confident.

In the initialization of the second learner (step 2 in the algorithm drawing), it was presented with a subset consisting of the 15 to 20% of the patterns whose confidence was lower than 0.3. This subset included most of the patterns belonging to the small clusters. It also had a small number of patterns from the main clusters. As the learner took into account all of the patterns, its decision boundary was a diagonal line from upper left to lower right. Thus, the difficult subset included data points at one vertical edge of each main cluster (and data points horizontally far from the centers of their gaussians). In the refining stage (step 3 in the algorithm drawing), the basic reliability coefficients for each learner were recalculated at each refining cycle, and then the data were split in a deterministic manner: each data point was assigned to the learner whose product of the confidence score on it and the reliability coefficient was higher. The refining stage had effect mostly on the first learner, which was able to produce a better estimation of the classification for the main gaussians.

In this example, the refining stage did not contribute much. We also

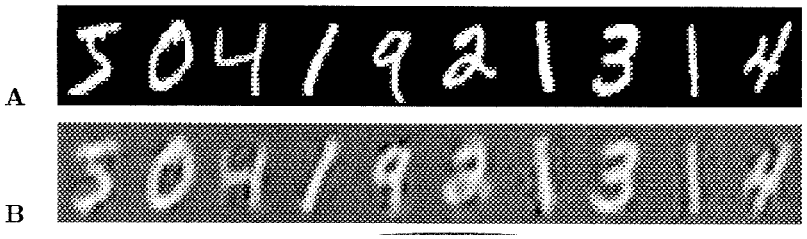

Figure 2: (A) Examples of digits from the NIST database. (B) Their representation by the first 32 principal components.

performed a slightly different variant of this problem in which the BME ensemble had a 6% error rate before refining, and after a few refining cycles it dropped to 4%. The first learner initially performed a compromise of the two separations, and when it had to perform only one separation, its performance improved.

7.2 Digit Recognition Results. The BME algorithm was empirically evaluated on digits from the NIST database (see Figure 2A). Preprocessing operations, similar to those described in Bottou et al. (1994), were applied to the digits. Digits were size normalized to fit a 20×20 pixel box (gray scale) centered within a 28×28 image. We then performed principal component analysis and used the first 32 components as input to our classifiers (see Figure 2B).

The basic classifier used was a feedforward neural network, trained via the backpropagation algorithm (with momentum). The network's input layer had 32 units, and its single hidden layer consisted of 16 units. The 10-dimensional output vector was used to extract the output digit and its confidence level.

In order to evaluate the unique contribution of the new algorithm, we compared it to a standard ensemble (*parallel machine*). This ensemble consisted of several learners trained independently, each with different starting conditions. The combination model used to extract the ensemble output was averaging of the output vectors of the different classifiers and decision according to the highest output. Increasing the number of networks improved the ensemble's performance.

We tested the performance ensembles trained with the BME algorithm. The initial training set for new learners added to the ensemble was constructed by choosing from the training set of each of the other learners those patterns on which it was less confident (we took $1/(a+b*n)$ of its set, where n is the current size of the ensemble and a, b are arbitrary constants). The confidence score of each pattern and a specific classifier was $(P_1 - P_2)^4$, where P_1 is the highest output of the classifier on the pattern and P_2 is its

Table 1: Performance of Various Ensembles on a Digit Recognition Task.

Number of Nets	Parallel Machine		Boosted Mixture of Experts		Multilevel Ensemble (2^*N nets)	
	Mean	SD	Mean	SD	Mean	SD
2	93.75%	0.35%	94.65%	0.3%	95.15%	0.3%
3	94.35	0.45	95.15	0.3	95.5	0.35
4	94.6	0.45	95.3	0.35	95.7	0.35
5	94.65	0.3	95.4	0.3	95.8	0.25
8	94.65	0.4	95.6	0.3	96	0.25
10	94.65	0.4	95.7	0.3	96.1	0.25

second highest output (probabilities were normalized to sum to 1 for any pattern). The gating function used at the refining step of the training, to get the probability of assigning a pattern to the training set of a specific classifier, was this confidence score (no global reliability coefficient was used). This gating function was also used in the combination model, as the weight given to each classifier in the weighted average of the output vectors.

We also performed a test of the multilevel ensemble. A simple average was applied to the output of two independently trained BME ensembles of N classifiers. Such an ensemble combines the advantages of an ensemble choosing the appropriate classifier for each pattern and an averaging ensemble.

Table 1 presents the performance of the three ensemble methods over a wide range of ensemble sizes. These results were collected using five different partitions of the data into a 49,000-digit training set and a 10,000-digit test set.

The basic MLP used had 32 inputs, 10 outputs, and 16 hidden units. By a naive counting, this gives $N = (32 + 1) * 16 + (16 + 1) * 10$, which is almost 700 free parameters. The effective number $\mathcal{N}$ is of the same order of magnitude. The naive number of parameters for both a parallel machine and a BME ensemble of k nets is kN, and for the multilevel ensemble it is $2kN$. Effectively, it is $\mathcal{N}$ parameters in the parallel machine, and for both the BME and the multilevel ensemble it is between $\mathcal{N}$ and $k\mathcal{N}$.

We tried to check whether the reported effect was due to only the increased number of parameters in the BME ensemble. The BME's number of parameters may be similar to that of a parallel machine, similar to that of a single classifier with a k-times larger hidden layer, or some intermediate case. For $k = 3$, the success rate of a parallel machine was 94.35%, the success rate for a larger net was 94.2%, and for a BME it was 95.15%. An average of two large nets had a success rate of 94.9%, while the multilevel ensemble had 95.5% success.

The results indicate that the performance of an ensemble machine trained

with the BME algorithm (and combined appropriately) is significantly better than a standard ensemble (parallel machine). The improvement rate is similar to that achieved using boosting (Drucker, Cortes, Jackel, Lecun, & Vapnik, 1994). It is encouraging that this improvement rate is kept even for a high number of classifiers (20% error reduction for 10 classifiers). The improved performance for a large ensemble was achieved despite the fact that the classifiers in this scheme were trained on a small portion of the data set. The improvement due to the BME algorithm beyond ensemble performance may be even larger when greater training sets are used (e.g., by multiplying samples using invariant transformations, as in Bottou et al., 1994).

The results further demonstrate the potential of combining the two basic schemes for ensemble machines in a multilevel approach. Our ensemble used a weighted average of classifiers, which tended to select the locally best classifier rather than average classifiers. Averaging the outputs of two such ensembles yielded further improvement in the results. These results are not fully contrasted with other ensembles of similar size, but when they are (two ensembles of 4 to 5 classifiers versus 8 to 10 classifiers) they have a slight advantage. Furthermore, because most studies claim that adding classifiers beyond a certain number is not expected to improve the performance further, the constant incremental improvement is encouraging.

8 Conclusions

This study analyzed two of the more advanced frameworks for ensembles of learning machines: boosting and the mixture of experts. We discussed the advantages and weaknesses of each algorithm and reviewed several ways in which the principles of these algorithms may be combined to achieve improved performance, including variants of each algorithm incorporating elements of the other.

We suggested a flexible procedure for constructing an ensemble machine based on principles of these two algorithms. The essential components are:

- Training several classifiers on subsets of the data with a significantly different distribution and using them in an ensemble.
- Dynamic classifier selection, which is common to the training and the test stages.
- Usage of a confidence measure for each of the classifiers as the gating function (in mixture-of-experts terminology), which determines their contribution to the ensemble output.

These principles lead to outperforming conventional ensemble machines.

The flexibility of the procedure is due mostly to the use of a confidence measure, which may be adjusted specifically for any classification or regression problem. This makes boostwise algorithms appropriate for regression problems as well. We further suggest an all-purpose confidence measure

by using a committee of simple learners as the basic learner in our algorithm. The disagreement among a committee for a given pattern becomes a confidence measure. We have made a distinction between two groups of ensemble machines: classifier selectors and classifier averagers. These two mechanisms provide different advantages for ensembles: using local experts may reduce bias, while averaging tends to reduce variance. We claim that a multilevel approach combining selection and averaging is capable of improving the performance of ensembles and that it may be better than a compromise between selection and averaging.

A digit recognition task from the NIST database was used to demonstrate the advantages of the BME and multilevel ensemble and achieve a significant reduction of the error rate over standard ensembles.

Acknowledgments

We thank NIST and H. Drucker for the handwritten digits database we used.

References

Bottou, L., Cortes, C., Denker, J., Drucker, H., Guyon, I., Jackel, L., LeCun, Y., Sackinger, U. M. E., Simard, P., & Vapnik, V. (1994). Comparison of classifier methods: A case study in handwritten digit recognition. In *Proceedings Int. Conf. on Pattern Recognition* (Vol. 12, pp. 77–82).

Breiman, L. (1996a). Bagging predictors. *Machine Learning, 24*, 123–140.

Breiman, L. (1996b). *Bias, variance and arcing classifiers* (Tech. Rep. TR-460). Berkeley: Department of Statistics, University of California, Berkeley.

Breiman, L. (1996c). Stacked regressions. *Machine Learning, 24*, 49–64.

Drucker, H., Cortes, C., Jackel, L., LeCun, Y., & Vapnik, V. (1994). Boosting and other ensemble methods. *Neural Computation, 6*(6), 1289–1301.

Drucker, H., Schapire, R., & Simard, P. (1993). Improving performance in neural networks using a boosting algorithm. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems, 5* (pp. 42–49). San Mateo, CA: Morgan Kaufmann.

Freund, Y. (1990). Boosting a weak learning algorithm by majority. In *3rd Annual Workshop on Computational Learning Theory* (pp. 202–216).

Freund, Y., & Schapire, R. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *2nd European Conference on Computational Learning Theory*.

Freund, Y., Seung, H., Shamir, E., & Tishby, N. (1993). Information, prediction and query by committee. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems, 5* (pp. 483–490). San Mateo, CA: Morgan Kaufmann.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*(10), 993–1001.

Ho, T., Hull, J., & Srihari, S. (1994). Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 16*(1), 66–75.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*(1), 79–87.

Jordan, M. I., & Jacobs, R. A. (1992). Hierarchies of adaptive experts. In J. E. Moody, S. J. Hanson, & R. P. Lippmann (Eds.), *Advances in neural information processing systems, 4* (pp. 985–992). San Mateo, CA: Morgan Kaufmann.

Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation, 6*(2), 181–214.

Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems, 7* (pp. 231–238). Cambridge, MA: MIT Press.

Meir, R. (1995). Bias, variance and the combination of least square estimators. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems, 7* (pp. 295–302). Cambridge, MA: MIT Press.

Perrone, M. P., & Cooper, L. N. (1993). When networks disagree: Ensemble method for neural networks. In R. J. Mammone (Ed.), *Neural networks for speech and image processing*. London: Chapman-Hall.

Raviv, Y., & Intrator, N. (1996). Bootstrapping with noise: An effective regularization technique. *Connection Science* (Special Issue), *8*, 356–372.

Schapire, R. (1990). The strength of weak learnability. *Machine Learning, 5*(2), 197–227.

Schwenk, H., & Bengio, Y. (1997). *Adaptive boosting of neural networks for character recognition* (Tech. Rep. TR-1072). Montreal: Department d'Informatique et Recerche Operationnelle, Université d'Montreal.

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query by committee. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory* (pp. 287–294).

Shimshoni, Y., & Intrator, N. (1996). Classifying seismic signals by integrating ensembles of neural networks. In S. Amari, L. Xu, L. W. Chan, I. King, & K. S. Leung (Eds.), *Proceedings of ICONIP Hong Kong. Progress in Neural Information Processing* (Vol. 1, pp. 84–90). New York: Springer-Verlag.

Tresp, V., & Taniguchi, M. (1995). Combining estimators using non-constant weighting function. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems, 7*. Cambridge, MA: MIT Press.

Waterhouse, S. R., & Cook, G. (1997). Ensemble methods for phoneme classification. In M. Mozer, J. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems, 9*. Cambridge, MA: MIT Press.

Waterhouse, S. R., & Robinson, A. J. (1996). Constructive algorithms for hierarchical mixtures of experts. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems, 8*. Cambridge, MA: MIT Press.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks, 5*, 241–259.

Received January 10, 1997; accepted December 10, 1997.