To appear in *the Proceedings of the International Joint Conference on Neural Networks*, pp xxx-xxx, Washington DC, 1999.

Decimated Input Ensembles for Improved Generalization

Kagan Tumer, NASA Ames Research Center, Caelum Research, Moffett Field, CA
Nikunj C. Oza, University of California, Berkeley, CA

Abstract— *Using an ensemble of classifiers instead of a single classifier has been demonstrated to improve generalization performance in many difficult problems. However, for this improvement to take place it is necessary to make the classifiers in an ensemble more complementary. In this paper, we highlight the need to reduce the correlation among the component classifiers and investigate one method for correlation reduction: input decimation. We elaborate on input decimation, a method that uses the discriminating features of the inputs to decouple classifiers. By presenting different parts of the feature set to each individual classifier, input decimation generates a diverse pool of classifiers. Experimental results confirm that input decimation combining improves generalization performance.*

I. Introduction

A classification learning task involves constructing a mapping from input instances (normally described by several features) to the (most likely) class to which the instances belong. Supervised learning consists of setting free parameters in the system (e.g., weights in a feed-forward neural network) according to a set of training examples–instances with known class memberships. The generalization performance of such a classifier (performance on patterns that have not previously been presented to the classifier) provides an indication of how well the classifier has learned the particular mapping [2]. However, in general, there are many reasonable generalizations given a finite training set. For example, when training a neural network classifier, different initial weights, learning rates, momentum terms, and architectures (e.g., number of hidden layers and hidden units, connections, single vs. distributed output encoding, etc.) affect how the classifier performs on novel examples [19]. For this reason, choosing a single classifier is not optimal. Even choosing the single best classifier among several classifiers trained using the same training examples is suboptimal because potentially valuable information may be wasted. These observations lead to the idea of generating classifier ensembles — also known as combiners or turnkey methods — to get better performance [12], [31], [32].

Recently, many researchers have demonstrated that using classifier ensembles (e.g., averaging the outputs of multiple classifiers before reaching a classification decision) leads to improved performance for many difficult generalization problems [4], [28], [32]. However, in many domains there are serious impediments to such "turnkey" classification accuracy improvements. Most notable among these is the deleterious effect of highly correlated classifiers on ensemble performance. One particular solution to this problem is training each classifier in the ensemble using a "new" training set generated by sampling from the original one. However, with finite number of patterns, this causes a reduction in the training patterns each classifier sees, often resulting in considerably worsened generalization performance for each individual classifier (particularly for high-dimensional data domains). Generally, this drop in the accuracy of the individual classifier performance more than offsets any potential gains due to combining, unless diversity among classifiers is actively promoted.

Therefore, in constructing the individual classifiers to be combined, it is important to have classifiers that have complementary information, i.e., have the lowest possible correlation [1], [14], [20], [30]. There are many methods for actively promoting diversity among the classifiers to be pooled, including bagging [5], [6], boosting [10], [11], cross-validation partitioning [18], [30], and error-correcting output codes [9].

In this work, we present *input decimation*, a method that:

- reduces the dimensionality of the data, thus lessening the impact of the "curse of dimensionality";
- reduces the correlation among the classifiers by training them on different features; and thereby,
- improves the classification performance of the ensemble.

In conventional dimensionality reduction methods such as Principal Component Analysis (PCA), the focus is on extracting the axes on which the data shows the highest variability. Although this approach "spreads" out the data in the new basis, and therefore is of great help in regression problems, there are no guarantees that the new axes are consistent with the discriminatory features in a classification problem. There are many variations on PCA that use local and/or use non-linear processing to improve dimensionality reduction [8], [17], [22], [23], [15], [16], [27], though they generally are based solely on the inputs. Input decimation, on the other hand, explicitly seeks out discriminating features, and eliminates input features that are least correlated with the outputs. The strength of the method is that only those features that have the most "explanatory" information pertinent to the discrimination task at hand are retained. As a result, this method is ideally suited for high dimensional data sets where the presence of redundant information is more likely.

Input decimation resulted in dimensionality reduction by a factor of four to six on the data sets we used in these experiments. Furthermore, this dimensionality reduction was accompanied by a drop on the error rate of input decimated combiners by 23-50% over combiners using the full feature set.

In the next section, we first briefly review a combining framework for classification problems that quantifies the need to reduce the correlation among individual classifiers (for more details, see [30]), and summarize the various correlation reduction methods. In Section III., we discuss the particular type of input decimation we implemented, i.e., how we chose the number of classifiers and the subsets of inputs for each classifier. In Section IV. we discuss experimental results on three data sets from the PROBEN1 benchmark [24] and the UCI Machine Learning Repository [3]. We conclude with an analysis of our results, and a discussion of the benefits of input decimation compared to other methods of constructing ensemble classifiers.

II. Background

It is well known that classifiers that have a distributed encoding for the output and that are trained to minimize a cross-entropy or mean square error (MSE) function over that output approximate the posterior probability densities of the corresponding classes [25], [26]. That is, the ith output of the classifier approximates the posterior probability density of the ith class. Therefore, we can model the ith output of such a classifier as follows(details of this derivation are in [29]):

$$f_i(x) = P(C_i|x) + \eta_i(x),$$

where $P(C_i|x)$ is the posterior probability distribution of the ith class given instance x, and $\eta_i(x)$ is the error associated with the ith output. Given an input x, if we only had this one classifier, we would classify x as being in the class i whose value $f_i(x)$ is largest.

Instead, if we use a combiner that calculates the arithmetic average over the outputs of N classifiers $f_i^m(x)$ ($m \in \{1, \ldots, N\}$), then we get an approximation to $P(C_i|x)$ as follows:

$$f_i^{ave}(x) = \frac{1}{N} \sum_{m=1}^{N} f_i^m(x) = P(C_i|x) + \bar{\eta}_i(x), \qquad (1)$$

where:

$$\bar{\eta}_i(x) = \frac{1}{N} \sum_{m=1}^{N} \eta_i^m(x)$$

and $\eta_i^m(x)$ is the error associated with the ith output of the mth classifier. The variance of $\bar{\eta}_i(x)$ is given by [30]:

$$\begin{aligned}\sigma^2_{\bar{\eta}_i} &= \frac{1}{N^2} \sum_{l=1}^{N} \sum_{m=1}^{N} cov(\eta_i^l(x), \eta_i^m(x)) \\ &= \frac{1}{N^2} \sum_{m=1}^{N} \sigma^2_{\eta_i^m(x)} + \frac{1}{N^2} \sum_{m=1}^{N} \sum_{l \neq m} cov(\eta_i^l(x), \eta_i^m(x)).\end{aligned}$$

If we express the covariances in terms of the correlations ($cov(x,y) = corr(x,y)\sigma_x\sigma_y$), assume the same variance $\sigma^2_{\eta_i}$ across classifiers, and use the average correlation factor among classifiers, δ_i, given by

$$\delta_i = \frac{1}{N(N-1)} \sum_{m=1}^{N} \sum_{l \neq m} corr(\eta_i^m(x), \eta_i^l(x)), \qquad (2)$$

then the variance becomes:

$$\sigma^2_{\bar{\eta}_i} = \frac{1}{N}\sigma^2_{\eta_i(x)} + \frac{N-1}{N}\delta_i \sigma^2_{\eta_i(x)}. \qquad (3)$$

Define b^{ave} to be the difference between the decision boundary of the average combiner and the true decision boundary between classes i and j. It can be shown [31] that the variance of this boundary offset is

$$\sigma^2_{b^{ave}} = \frac{\sigma^2_{\bar{\eta}_i} + \sigma^2_{\bar{\eta}_j}}{s^2}$$

where s is the difference between the derivatives of the posteriors of classes i and j. Using equation (3) and some simplification, we get:

$$\sigma^2_{b^{ave}} = \frac{\sigma^2_{\bar{\eta}_i} + \sigma^2_{\bar{\eta}_j}}{Ns^2} + \frac{N-1}{Ns^2}\Big(\delta_i \sigma^2_{\eta_i^2(x)} + \delta_j \sigma^2_{\eta_j^2(x)}\Big).$$

Using the fact that the errors across classes are independent and identically distributed (i.e., $\sigma_{\eta_i(x)} = \sigma_{\eta_j(x)}$) leads to:

$$\sigma^2_{b^{ave}} = \frac{\sigma^2_b}{N}\left(1 + (N-1)\frac{\delta_i + \delta_j}{2}\right). \qquad (4)$$

where σ^2_b is the variance of the boundary offset for a single component of the average combiner. The correlation term above applies only to classes i and j. To extend this expression to include all the classes, we use the following:

$$\delta = \sum_{i=1}^{L} P_i \delta_i \qquad (5)$$

where P_i is the prior probability of class i. We can now write the average combiner's added error beyond the Bayes error rate as:

$$\begin{aligned} E^{ave}_{add} = \frac{s}{2}\sigma^2_{b^{ave}} &= \frac{s}{2}\sigma^2_b\left(\frac{1+\delta(N-1)}{N}\right) \\ &= E_{add}\left(\frac{1+\delta(N-1)}{N}\right). \qquad (6) \end{aligned}$$

From this, we can immediately see that as the correlations among classifiers decrease, the added error introduced by the average combiner decreases, which leads to improved combiner performance.

III. Input Decimated Features

Many combining methods incorporate (explicitly or implicitly) a correlation reduction method to improve generalization. Some partition the training set much like one does when using cross-validation and train one classifier on each partition [18], [30]. A better known method, *bagging* [5], [6], constructs several sets of m training examples drawn randomly with replacement out of the original set of m training examples and trains one classifier using each of these resampled training sets. *Boosting* [10], [11] also subsamples the input space, but the training samples are selected based on their "difficulty" to be learned and the training sets are constructed iteratively.

Unlike the methods mentioned above, the approach we adopt here is based on using different subsets of *input features*, rather than a different subset of *input patterns*. Intuitively, input decimation decouples the classifiers by exposing them to different aspects of the same data. In this method one trains L classifiers, one corresponding to each class in an L-class problem. For each classifier, one selects a subset of the input features according to their correlation to the corresponding class. The objective is to "weed" out all input features that do not carry discriminating information relevant to the particular class. Cherkauer reports improved generalization using a similar method in which hand-tuned feature sets are used to train classifiers in a combiner [7]. The main advantage of input decimation over standard dimensionality reduction methods such as Principal Component Analysis (PCA) is that input decimation selects features based on their correlation with the outputs.

In this paper, we report the results of experiments in which each decimated feature set had the same dimensionality (e.g., we chose a fixed number of highest-correlation inputs for each classifier).[1] The base classifiers we selected for this study are Multi-Layered Perceptrons (MLPs) trained with the backpropagation algorithm [13]. For comparison purposes, we also report the results of using a fixed number of highest-correlation principal components for each classifier. We also report results of base classifiers trained on the full feature set, along with results from combiners that pool L base classifiers.[2] We have selected the "averaging" combiner, as it is computationally cheap and provides results quantitatively similar to combiners discussed above (e.g., bagging, boosting etc.) [29].

IV. Experimental Results

A. Data Sets and Classifiers

The Gene data has 120 input features and three class variables [21], [24]. Through experimentation,[3], we selected an MLP with a single hidden layer of 20 units, a learning rate of 0.5 and a momentum term of 0.8.[4]

The Splice data consists of 60 input features and three classes [3]. Here we selected an MLP with a single hidden layer composed of 120 units, a learning rate of 0.05, and a momentum term of 0.1. The Satellite

[1]We are currently exploring cases where each classifier has (potentially) access to a different number of inputs.

[2]These combiners will be referred to as "original combiners" in the remainder of this paper to distinguish them from the decimated combiners.

[3]The learning momentum rates were selected experimentally, whereas the number of hidden units was selected through cross-validation.

[4]For each data set, we chose the same number of hidden units, learning rate, and momentum term to use in training both the isolated neural networks and the networks that were part of combiners.

Image data has 36 input features and 6 classes [3]. We chose an MLP with a single hidden layer of 50 units, and a learning rate and momentum term of 0.5. In all three data sets, we performed 20 independent training runs of 100 epochs each for both the single networks and the combiners.

The average classification accuracies (percentage correct) with standard deviations over 20 runs of the original MLP ("Single") and the original combiner ("Average") and the average correlations among the components in the combiners are given in Table I. For each problem there as many classifiers in the combiners as there are classes in the data (e.g., we combined 3 classifiers for Gene and Splice and 6 for Satellite). Note that even though the networks in the combiner are structurally the same and are trained on precisely the same data, they are not perfectly correlated due to the random initial assignment of weights to these networks. This is the main reason why we get some improvement in the classification accuracy through ensembles. For the Gene data, the average combiner was significantly more accurate than the single MLP, while for the Satellite Image and Splice data sets, the combiner was only marginally more accurate.

TABLE I
Average Accuracy of Original Network and Combiners

	Single	Average	Corr.
Gene	83.417 ± .796	86.418 ± .342	.7910
Splice	84.722 ± .534	85.372 ± .631	.7263
Satellite	87.785 ± .685	89.010 ± .273	.9523

B. Input Decimation Combining

This section describes experiments in which we chose a fixed number of inputs or principal components for each of the classifiers in our ensemble. For the Gene data we chose a range from 20 inputs to 110 inputs (in increments of 10) out of the original 120. For the Splice data, we chose a range from 10 inputs up to 50 inputs (in increments of 10) out of the original 60. For the Satellite Image data, we chose a range from 9 to 27 inputs (in increments of 9) out of the 36 total inputs. For input decimation, for each classifier, we chose the features having the highest correlations with the corresponding class output. The accuracies of the resulting classifiers with standard deviations over 20 independent runs are given in Tables II-IV below, for the input Decimated Features (DF) and PCA. The column "Single" is the average classification accuracy of the individual components in the decimated combiners.

In case of the Gene data, the average combiners with 20, 30, and 40 inputs are significantly more accurate than both the original network combiners described in the previous section and their PCA counterparts. Note also that the performances of the PCA-based combiners vary arbitrarily as the number of principal components changes, while the performances of the feature-based combiners are more stable. This is consistent with the fact that principal components are not necessarily good discriminative features, and eliminating particular principal components have unpredictable effects on the classification performance.

In the Splice data experiments, *all* the decimated feature-based combiners significantly outperformed both the original combiner and the PCA-based combiners. What is particularly notable in this case is that a reduction of dimensionality based on PCA has a strong negative impact on the classification performance. With 20 principle components for example, the

TABLE II
Gene Data: Influence of Dimensionality on Combiner Performances

Dim.		Single	Average	Corr.
110	DF	83.636 ± 0.930	86.482 ± 0.851	0.800
	PCA	76.595 ± 1.086	85.876 ± 0.529	0.394
100	DF	83.623 ± 1.165	86.419 ± 0.731	0.791
	PCA	76.166 ± 0.561	85.574 ± 0.837	0.457
90	DF	82.947 ± 1.041	86.091 ± 0.584	0.788
	PCA	81.761 ± 1.222	85.839 ± 0.885	0.729
80	DF	83.632 ± 1.216	86.457 ± 1.015	0.794
	PCA	83.316 ± 0.894	86.368 ± 0.530	0.781
40	DF	84.237 ± 0.897	87.276 ± 0.671	0.805
	PCA	65.737 ± 2.141	80.958 ± 0.806	0.240
30	DF	83.422 ± 0.836	88.045 ± 0.617	0.762
	PCA	76.784 ± 1.645	84.767 ± 0.919	0.523
20	DF	85.754 ± 0.955	89.546 ± 0.548	0.734
	PCA	67.192 ± 0.905	83.001 ± 0.697	0.665

TABLE III
Satellite Image Data: Influence of Dimensionality on Combiner Performances

Dim.		Single	Average	Corr.
27	DF	86.512 ± 0.764	86.482 ± 0.851	0.923
	PCA	87.863 ± 0.572	88.820 ± 0.154	0.897
18	DF	82.645 ± 1.164	86.419 ± 0.731	0.856
	PCA	84.877 ± 1.031	89.510 ± 0.242	0.910
9	DF	70.679 ± 0.838	86.091 ± 0.584	0.395
	PCA	83.574 ± 0.756	89.035 ± 0.252	0.948

TABLE IV
Splice Data: Influence of Dimensionality on Combiner Performances

Dim.		Single	Average	Corr.
50	DF	85.152 ± 0.619	86.896 ± 0.312	0.857
	PCA	83.230 ± 0.868	85.014 ± 0.767	0.861
40	DF	86.460 ± 0.607	88.532 ± 0.523	0.855
	PCA	82.286 ± 0.824	84.939 ± 0.556	0.838
30	DF	87.880 ± 0.928	90.329 ± 0.833	0.859
	PCA	81.276 ± 0.726	84.073 ± 0.355	0.805
20	DF	88.310 ± 0.666	92.380 ± 0.714	0.792
	PCA	79.263 ± 0.548	82.493 ± 0.495	0.785
10	DF	84.669 ± 0.561	92.342 ± 0.737	0.719
	PCA	78.109 ± 0.542	80.066 ± 0.400	0.816

performance of the individual classifiers drops by 7 %, whereas the performance of the DF individual classifier increases by 3 %. The improvement of the performance due to decimation is an initially surprising aspect of these experiments. However, an analysis shows that the inputs that were decimated were in fact providing "noise" to the classifier. Therefore, although it is theoretically possible for the classifier with all the features to do well, it is in practice very difficult to train.

In the Satellite Image data however, the input decimated combiner with 27 features was the only one that did not perform significantly worse than the single neural network and the original combiner. This is the data set with the lowest dimensionality, and shows two things: (i) in order to take advantage of input decimation, the initial dimensionality has to be very high; and (ii) If there are features that have significant meaning, they need to be included in the feature set regardless of their correlation to the particular output. We observed that consecutive groups of four features in the satellite image data set correspond to spectral values for a given pixel. In examining the eigenvalues and eigenvectors, we found that the highest eigenvalue was 91.6% of the sum of the eigenvalues, and the corresponding eigenvector was a simple linear combination of the four spectral values across all the pixels. In this case, the higher principal components are good discriminative features. Because all the features have the same range, directions of higher variability are more likely to correspond to good discriminative features.

V. Analysis

We can analyze the performance of the decimated feature combiners in more detail by examining the average performances of the individual components in the combiners (column "Single") and the average correlations among the components in Tables II-IV. The correlation among the individual classifiers was reduced in all cases but the Splice data. Furthermore, only for the Gene data did this drop come without a substantial performance penalty. For the Gene data, as more features were removed, the accuracy of the individual classifiers decreased, but the correlation among the classifiers also decreased. This trade-off needs to be balanced in order to find the best combiner. In case of the Satellite Image data, as the number of inputs decreased, even though the correlations among the classifiers decreased, the performance of the individual classifiers dropped sufficiently to eliminate any potential benefits. For the Splice data, as the number of inputs decreased, the correlation decreased noticeably, while the individual classification rate improved, thereby yielding superior combiners.

As discussed above, there is some variation in how well input decimation works. In cases where the correlation drop is not accompanied by a correspondingly large drop in individual performance, gains can be achieved through combiners. In some cases however, instead of a drop, a reduction in correlation is coupled with improved individual classification performance. We attribute this phenomenon to the reduction in the complexity of the model required. Therefore, reduction of overfitting can be a side benefit of input decimation in many cases. (This is akin to a signal-to-noise issue. Removing features with low correlation to the outputs is removing noise from the classifier.)

As the experiments show, input decimation performed substantially better than the conventional combiners on two of the data sets. In the Satellite Image data, on the other hand, input decimation failed to yield improvements over conventional algorithms (indeed failed to reach their level in many cases). According to the documentation for this data [3], the features (17-21) corresponding to the central pixel (the center of the image) are more useful than the others; however, many of the other inputs have higher correlations with the different classes. Because of this, many of the classifiers had to have a large number of features, before features 17-21 were included. In this case, due to a particular coding of the features, certain features had to be included in all cases.

VI. Conclusion

This paper discussed the importance of reducing the correlation among classifiers in an ensemble and explored input decimation as a technique for correlation reduction. Experimental results on three data sets revealed several benefits of input decimation over using combiners based on dimensionality reductions relying on PCAs.

Furthermore, training individual classifiers with reduced input sets gives our method several advantages over other methods of introducing diversity in a pool. In particular, input decimation and subsequent training of classifiers on smaller features sets:

- reduces overfitting in each classifier;
- reduces training times for the individual classifiers;
- reduces the correlations among the classifiers; and
- provides more insight into how each classifier and associated combiner makes its decisions (i.e., which features carry the most discriminating features for each task.)

The experiments reported in this article were based on all classifiers being trained on the *same* number of features. We are currently investigating allowing a variable number of inputs for each classifier. We are also investigating various measures that reveal the best set of variables to include in each classifier. Such measures need to account for the correlations (or other measure of similarity) of the inputs with each class output, the correlations among the inputs themselves, and the number of inputs common to the classifiers. We are also exploring "prelearning" (i.e., preparing the components before learning by training them only to minimize their correlations with each other). This approach shifts the component classifiers to different parts of the parameter space so that they are less likely to find the same locally-optimal solution during training.

Acknowledgments. Part of this work was done while Nikunj Oza was visiting NASA Ames Research Center. He was also supported by a Schlumberger Foundation Fellowship.

References

[1] K. M. Ali and M. J. Pazzani. On the link between error correlation and error reduction in decision tree ensembles. Technical Report 95-38, Department of Information and Computer Science, University of California, Irvine, 1995.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

[3] C. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases, 1998. (URL: http://www.ics.uci.edu/~mlearn/MLRepository.html).

[4] L. Breiman. Stacked regression. Technical Report 367, Department of Statistics, University of California, Berkeley, 1993.

[5] L. Breiman. Bagging predictors. Technical Report 421, Department of Statistics, University of California, Berkeley, 1994.

[6] L. Breiman. Bias, variance and arcing classifiers. Technical Report 460, Department of Statistics, University of California, Berkeley, 1996.

[7] K. J. Cherkauer. Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. In *Working Notes of the AAAI Workshop on Integrating Multiple Learned Models*, pages 15–21, 1996.

[8] D. de Ridder and R. P. W. Duin. Sammon's mapping using neural networks: A comparison. *Pattern Recognition Letters*, 18:1307–1316, 1997.

[9] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

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

[11] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[12] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1000, 1990.

[13] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Macmillan, New York, 1994.

[14] Robert Jacobs. Method for combining experts' probability assessments. *Neural Computation*, 7(5):867–888, 1995.

[15] N. Kambhatla and T. K. Leen. Fast non-linear dimension reduction. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems-6*, pages 152–153. Morgan Kaufmann, 1994.

[16] N. Kambhatla and T. K. Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9:1493, 1997.

[17] D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the 13th International Conference on Machine Learning*, pages 284–292, 1996.

[18] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation and active learning. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems-7*, pages 231–238. M.I.T. Press, 1995.

[19] E. Levin, N. Tishby, and S. A. Solla. A statistical approach to learning and generalization in layered neural networks. *Proc. IEEE*, 78(10):1568–74, Oct 1990.

[20] R. Meir. Bias, variance, and the combination of estimators; the case of least linear squares. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems-7*, pages 295–302. M.I.T. Press, 1995.

[21] M. O. Noordewier, G. G. Towell, and J. W. Shavlik. Training knowledge-based neural networks to recognize genes in DNA sequences. In R.P. Lippmann, J.E. Moody, and D.S. Touretzky, editors, *Ad. in Neural Information Processing Systems-3*, pages 530–536. Morgan Kaufmann, 1991.

[22] E. Oja. *Subspace Methods of Pattern Recognition.* Research Studies Press, Letchworth, England, 1983.

[23] E. Oja. Priciple components, minor components, and linear neural networks. *Neural Networks*, 5:927–936, 1992.

[24] Lutz Prechelt. PROBEN1 — A set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, D-76128 Karlsruhe, Germany, September 1994. Anonymous FTP: /pub/papers/techreports/1994/1994-21.ps.Z on ftp.ira.uka.de.

[25] M.D. Richard and R.P. Lippmann. Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Computation*, 3(4):461–483, 1991.

[26] D. W. Ruck, S. K. Rogers, M. E. Kabrisky, M. E. Oxley, and B. W. Suter. The multilayer Perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298, 1990.

[27] J.W. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE transactions on Computers*, 18:401–409, 1969.

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

[29] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, February 1996.

[30] K. Tumer and J. Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3 & 4):385–404, 1996.

[31] K. Tumer and J. Ghosh. Linear and order statistics combiners for pattern classification. In A. J. C. Sharkey, editor, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, pages 127–162. Springer-Verlag, London, 1999.

[32] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.