Dimensionality Reduction Through Classifier Ensembles

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

September 17, 1999

Abstract

In data mining, one often needs to analyze datasets with a very large number of attributes. Performing machine learning directly on such data sets is often impractical because of extensive run times, excessive complexity of the fitted model (often leading to overfitting), and the well-known "curse of dimensionality." In practice, to avoid such problems, feature selection and/or extraction are often used to reduce data dimensionality prior to the learning step. However, existing feature selection/extraction algorithms either evaluate features by their effectiveness across the *entire* data set or simply disregard class information altogether (e.g., principal component analysis). Furthermore, feature extraction algorithms such as principal components analysis create new features that are often meaningless to human users. In this article, we present input decimation, a method that provides "feature subsets" that are selected for their ability to discriminate among the classes. These features are subsequently used in ensembles of classifiers, yielding results superior to single classifiers, ensembles that use the full set of features, and ensembles based on principal component analysis on both real and synthetic datasets.

1 Introduction

In data mining, one often deals with large datasets with a high number of input attributes [14, 23, 25]. Performing machine learning directly on such datasets is typically impractical for a multitude of reasons. Generally, for such data sets:

- Learning algorithms are slow due to the large number of parameters that need to be learned;

- Many attributes are irrelevant for the task at hand, resulting in wasted effort, overfitting, or worse, learning spurious relationships; and
- The number of training examples needed to produce a meaningful model over the full attribute space is prohibitively large—this is known as the "curse of dimensionality" [7].

To alleviate at least some of these problems, feature selection or feature extraction is often used prior to learning. Feature selection is the act of choosing a subset of the original features according to some criterion for deciding how relevant each feature is for the task at hand.[1] However, these methods, when applied to classification problems, typically choose features according to the criterion of how useful they are at discriminating among *all* classes, or simply choose features that have high variability with little or no regard for their discriminatory power. In many real datasets, however, there are features that are very useful at distinguishing one class from the remaining classes. Feature extraction involves calculating new features from the original ones with the intent of keeping the "salient information" while reducing the dimensionality of the data [63], often resulting in new features that are not intuitively understandable. Furthermore, many unsupervised feature extraction methods such as Principal Components Analysis (PCA) disregard class information and, therefore, are not suited for finding features that are useful for classification.

In this paper, we present input decimation, a method that chooses different subsets of the original features for use in classifiers that are part of an ensemble. This method not only reduces the dimensionality of the data, but uses this dimensionality reduction to reduce the correlation among the classifiers in an ensemble, thereby improving the classification performance of the ensemble [58, 61] (the relationship between ensemble performance and correlation among its components has been extensively discussed [2, 31, 43, 59]). In this article, we present details of this method, along with extensive simulations on both real and synthetic data sets showing that input decimation reduces the error up to 90 % over single classifiers, ensembles trained on full features and ensembles trained on principal components. Note that in this study we use the "averaging" ensemble to compare ensembles *with* and *without* input decimation, rather than compare input decimation to other more sophisticated methods. Indeed, ensemble methods such as bagging, boosting, and stacking (discussed in Section 2) can be used *in conjunction* with input decimation. In that sense, input decimation is orthogonal to those methods. In this study, we select the averaging ensemble because, due to its simplicity, it provides a clear comparison of the results with and without input decimation.

[1] In this article we restrict attention to classification problems.

In Section 2, we briefly review known methods for dimensionality reduction and ensemble methods, and discuss an ensemble framework that quantifies the need for correlation reduction among classifiers (see [59] for further details). In Section 3 we present input decimation, and in Section 4 we provide experimental results on three data sets from the PROBEN1 benchmark [51] and the UCI Machine Learning Repository [8], along with several synthetic datasets. We conclude with a discussion of the benefits and limitations of input decimation and highlight directions for future research.

2 Background

As we mentioned above, input decimation uses dimensionality reduction to reduce the correlation among classifiers in an ensemble, yielding superior ensemble classifier performance. Because input decimation is a both a dimensionality reduction method and an ensemble method, below we present a brief background for both. Furthermore, to emphasize the connection between these two concepts, we summarize a framework that shows that reducing the correlation among classifiers (e.g., through input decimation) in an ensemble improves classification performance.

2.1 Dimensionality Reduction

Most of the known dimensionality reduction methods are examples of one of two different classes of methods: feature selection and feature extraction. In feature selection one chooses some criterion (e.g., statistical correlation or mutual information) for deciding how relevant each feature is for the classification or regression task and chooses some subset of the features according to this criterion [3, 9, 10, 19, 32, 40]. In *filter* methods for feature selection, the data with the chosen subset of features is then presented to a learning algorithm. In *embedded* methods, feature selection is done as part of the learning algorithm. Decision-tree learning (e.g., [52]) is one example in which an embedded feature selection method is used—attributes are chosen based on information gain at each node in the decision tree. In *wrapper* methods, the learning algorithm itself is run with various subsets of features and the learner that performs best is chosen [37]. However, most of these feature selection methods attempt to choose features that are useful in discriminating across all classes. One exception is to break an L-class problem into $\frac{L!}{2\,(L-2)!}$ two-class problems and performs feature selection within each of those problems [39]. In many real-world problems, there are features that are useful at distinguishing whether an instance is of one particular class but are not useful at distinguishing among the remaining classes. Most feature selection algorithms also

choose individual features in a greedy manner, i.e., they do not account for the interactions among various sets of features. Methods that attempt to overcome that (e.g., [38]) are computationally more expensive, a problem that is accentuated by large datasets.

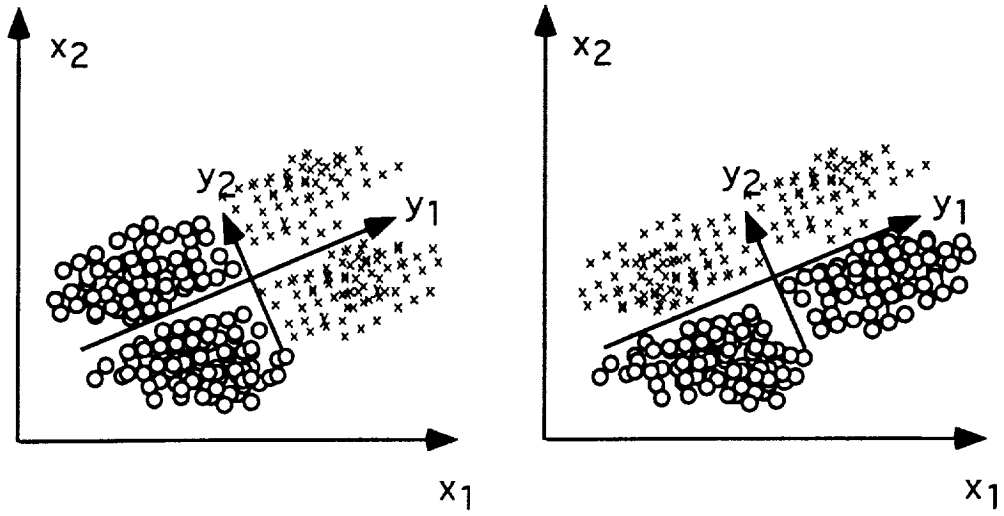

Figure 1: PCA and classification: The first principal can provide a good discriminating feature (left) or a poor one (right), since the class membership information is not used.

Feature extraction algorithms such as Principal Components Analysis (PCA) [7, 33, 48] or Independent Component Analysis (ICA) [30] reduce the dimensionality of the data by creating new features. Linear PCA, perhaps the most commonly used feature extraction method, creates new uncorrelated features that are linear combinations of the original features. The aim of PCA is to find the set of features on which the data shows highest variability. However, it is generally difficult to intuitively understand these new features. Furthermore, PCA gives high weight to features with higher variabilities whether they are useful for classification or not. In other words, because unsupervised feature extraction methods such as PCA do not use the class labels to create the new features, they often yield features that are not useful for classification [7]. Figure 1 demonstrates the perils of not using class information. The left half of the figure shows a case in which PCA works effectively. In this case the first principal component corresponds to the variable with the highest discriminating power. The right half shows a similar dataset (similar data distribution and linearly separable). However, because the first principal component is not "aligned" with the class labels, selecting this component is a poor choice for this problem. Indeed, an input set consisting of only the first component would provide practically random decisions on this data set. These examples show that using PCA for classification problems is a dangerous process, as there is little information to

determine the amount of discriminating information that is kept in the principal components that account for most of the variability in the input data.

There are variations on PCA that use local and/or nonlinear processing to improve dimensionality reduction [16, 35, 36, 46, 47, 56]. One such method uses vector quantization to create several cells, and performs PCA within each cell [35]. Each example is then coded using the principal components for the closest cell. Although these methods implicitly account for some class information and therefore are better suited than global PCA methods for classification problems, they do not directly use class information.

2.2 Ensembles and Correlation

2.2.1 Ensemble methods

A classification task consists of determining the class membership of a pattern, based on an input vector consisting of features describing that pattern. Learning generally involves using training examples—patterns with known class memberships—to construct a classifier that generalizes, i.e., responds correctly to novel patterns. However, in general, there are many possible generalizations based on a finite training set [41]. For example, when training a feed forward 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. For this reason, choosing a single classifier, even the "best" classifier in terms of generalization error, is not necessarily optimal, because potentially valuable information may be discarded. This observation leads to the idea of classifier ensembles, where the outputs of multiple classifiers are "pooled" before a class label is assigned [11, 26, 62]. In constructing an ensemble, two issues arise: the method by which the outputs are combined, and the method by which the individual classifiers are constructed. (See [17, 57] for a review of ensemble methods.)

Majority voting is one of the most basic methods of combining [4, 26]. If the classifiers provide probability values, simple averaging is an effective ensemble method and has received a lot of attention [42, 50, 59]. Weighted averaging has also been proposed and different methods for computing the weights of the classifiers have been examined [6, 27, 31, 34, 42, 44]. Such linear combining techniques have been mathematically analyzed in depth [12, 27, 50, 59]. Other non-linear ensemble schemes include rank-based combining [1, 29], belief-based methods [54, 64, 65], and order-statistic ensembles [60].

In constructing the individual classifiers to be combined, many methods are used, including simply training all classifiers as if they were stand-alone classifiers and then combining

them into an ensemble. However, one can also try to actively promote some diversity among the classifiers (we elaborate on the reasons for this in the next section). One such method partitions the training set much like one does when using cross-validation and trains one classifier on each partition [28, 59]. Another method, known as *bagging* [13], 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* [24] is similar to bagging, except that the process of drawing training examples and constructing classifiers is done iteratively [21, 22, 24]. A probability distribution on the training examples is maintained and training sets are drawn with replacement according to this distribution. After a classifier is constructed, the probability distribution is adjusted so that examples that were misclassified are more likely to be chosen in the next iteration than examples that were correctly classified. Another way of constructing a set of complementary classifiers is to give each classifier a different output target. One method is error-correcting output coding [18]. In this method, the set of classes is randomly partitioned into two subsets (A_l and B_l) T times (that is $l \in \{1, 2, \ldots, T\}$), and each of the T classifiers is assigned one partition. The lth classifier's copy of the training examples is relabeled as follows: the example is considered positive if the class of that example is in B_l and negative otherwise. Of course, because the data is relabeled differently for each classifier, each classifier will be different. Each of these methods relies on reducing the correlations among the classifiers that are part of an ensemble. We now summarize a classification framework that explicitly connects the reduction in the classification error of an ensemble to the correlation among the constituent classifiers in that ensemble.

2.2.2 The Need for Correlation Reduction

In this article we focus on classifiers that model the *a posteriori* probabilities of output classes. Such algorithms include Bayesian methods, and properly trained feed forward neural networks [53, 55]. Therefore, we can model the ith output of such a classifier as follows (details of this derivation are in [58, 59]):

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

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 have one classifier, we classify x as being in the class i whose value $f_i(x)$ is largest.

Instead, if we use a ensemble 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:

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

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.

Now, the variance of $\bar{\eta}_i(x)$ is given by [59]:

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

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

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

then the variance becomes:

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

Based on this variance, we can compute the variance of the decision boundary and, generalizing this result to the classifier error, we obtain the relationship between the error of the ensemble and that of an individual classifier:

$$E_{add}^{ave} = \left(\frac{1 + \delta(N-1)}{N}\right) E_{add} \tag{4}$$

where

$$\delta = \sum_{i=1}^{L} P_i \delta_i \tag{5}$$

with P_i is the prior probability of class i.

Equation 4 quantifies the connection between error reduction and the correlation among the errors of the classifiers. This result leads us to seek to reduce the correlation among classifiers prior to using them in an ensemble. In the next section we present the input decimation algorithm which merges dimensionality reduction and correlation reduction to provide classifier ensembles.

3 Input Decimation

Unlike methods such as bagging and boosting which work by using subsets of *input patterns*, input decimation focuses on subsets of *input features*. 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 much discriminating information relevant to the particular class.

Our learning algorithm works as follows:

- Convert the training dataset to a distributed encoding if necessary.
- For each class $i \in \{1, 2, \ldots, L\}$,
  - Compute the correlation between each feature and y_i the output for class i.
  - Select n_i features having the highest correlation with the class i output. Call this set of features F_i.
  - use a learning algorithm to realize the mapping from each new feature set (F_i) to the full outputs.

Given a new example x, we classify it as follows:

- For each learning algorithm f^m in the ensemble ($m \in \{1, 2, \ldots, L\}$),
  - Calculate the output $f_i^m(x)$ for each class i.
- For each class $i \in \{1, 2, \ldots, L\}$,
  - Calculate the ensemble average of $f_i^m(x)$ for all $m \in \{1, 2, \ldots, L\}$, yielding $f_i^{ave}(x)$.
- Return the class $i = argmax_i f_i^{ave}(x)$.

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. Cherkauer uses a similar feature selection method, but the feature subsets are selected by hand [15], whereas Bay proposes a method where the subsets are selected at random [5]. In this paper, we report results on real datasets in which each decimated feature set had the same dimensionality (i.e., we chose a fixed number of highest-correlation inputs for each classifier) as well as results with decimated feature sets of different dimensionality. We also present controlled experiments on synthetic datasets.

As mentioned earlier, input decimation reduces correlation among individual classifiers by using different subsets of input features, while methods such as bagging and boosting

reduce correlation by choosing different subsets of input patterns. These facts imply that input decimation is orthogonal to pattern-based methods such as bagging and boosting, i.e., one can use input decimation in conjunction with those methods. We will elaborate on this point in Section 5.

4 Experimental Results

In this section, we present the results based on input decimation on several synthetic and real datasets. In these experiments, for an L-class problem, we train L classifiers[2], each of which uses some of the features having highest correlation with the presence or absence of one particular class. The results given in the tables are percentages correct and standard error on the test set averaged over 20 independent runs.

As a standard against which to compare our input decimation results, we also trained a classifier on the full feature set (referred to as the "single classifier") and separately trained L copies of the same classifier and incorporated them into an ensemble average (referred to as the "original ensemble").

4.1 Synthetic Data

We tested input decimation on the following six synthetic datasets.

- Set 1:
  - Three classes–one unimodal Gaussian per class.
  - 300 training patterns and 150 test patterns–100/50 per class.
  - 100 features per pattern where:
    - 10 relevant features per class–each class's peak is a multivariate normal distribution in 10 independent dimensions distributed as $N(40, 5^2)$. There are no features in common among the three classes' peaks. Therefore, there are 30 relevant features.
    - 70 irrelevant features–distributed as $U[-100, 100]$.
- Set 2: Same as Set 1, except that 50 irrelevant features were added to the 30 relevant features, for a total of 80 features in the dataset.

[2]In this article we use multi-layered perceptrons (MLP) trained with the backpropagation algorithm as our classifiers. The learning rates, momentum terms, and stoppage times were chosen experimentally, whereas the number of hidden units was selected using cross-validation.

- Set 3: Same as Set 1, except that only 20 irrelevant features were added to the 30 relevant features, for a total of 50 features in the dataset.
- Set 4: Same as Set 1, except that there are 1000 training examples and 500 testing examples per class–a total of 3000 training examples and 1500 testing examples.
- Set 5: Same as Set 1, except that there is overlap among the relevant features for each class (e.g., classes one and two have three relevant features in common).

In the next subsections, we present our results for each dataset followed by our analysis. Table 1 provides the classification performance for single classifiers and ensembles on the full feature set[3], along with the correlations among the individual classifiers in the ensemble. Note that the original ensembles always give some improvement over the individual classifiers in each case. In Tables 2-6, we provide the single classifier and ensemble results when only subsets of the feature set are used. The first column provides the dimensionality of the data (number of features per classifier), the second column specifies which dimensionality methods was used (input decimation or PCA), and the last column provides the average correlation among the classifiers in the ensemble.

Table 1: Single Classifier and Ensemble Performance on the Full Feature Set

	Single	Ensemble	Corr.
Set 1	84.267 ± .2.9394	88.333 ± 1.9720	.678
Set 2	83.467 ± 3.1241	89.600 ± 2.0374	.706
Set 3	84.633 ± 2.8005	89.500 ± 2.0535	.726
Set 4	90.480 ± 0.6849	93.393 ± 0.4948	.808
Set 5	78.5 ± 2.3273	84.633 ± 2.3710	.676

4.1.1 Set 1

Table 2 presents the results for the first data set.[4] Input decimation provided the best performance for subsets with 20 and 30 features. This is consistent with the data as there are 30 relevant features, out of which at least 10 are needed for each classifier. The 5 and 10 feature ensembles also performed fairly well, even though the single component classifiers

[3]The ensemble consists of 3 classifiers for all data sets.

[4]The single classifier used was an MLP with a single hidden layer consisting of 95 units, trained using a learning rate of 0.2 and a momentum term of 0.5.

performed poorly. In these cases, there is very little correlation among the individual classifiers, which accounts for the substantial improvements in performance due to the ensemble (see Equation 4).

Table 2: Synthetic Dataset 1: Influence of Dimensionality on Ensemble Performances

Dim.		Single	Ensemble	Corr.
70	DF	86.911 ± 2.157	91.733 ± 1.467	0.751
	PCA	86.422 ± 2.689	91.133 ± 1.634	0.769
60	DF	87.678 ± 2.510	92.333 ± 1.844	0.759
	PCA	85.778 ± 2.252	90.867 ± 1.416	0.754
50	DF	89.500 ± 2.112	93.200 ± 1.470	0.783
	PCA	86.467 ± 2.409	91.300 ± 1.542	0.764
40	DF	90.189 ± 1.865	93.4 ± 1.133	0.823
	PCA	86.744 ± 2.162	91.700 ± 0.954	0.787
30	DF	91.322 ± 1.911	95.233 ± 0.8505	0.811
	PCA	86.456 ± 2.566	90.733 ± 1.685	0.765
20	DF	85.756 ± 2.523	95.033 ± 1.570	0.638
	PCA	86.445 ± 2.093	91.100 ± 1.480	0.784
10	DF	66.989 ± 3.165	93.967 ± 2.005	0.130
	PCA	85.656 ± 2.211	90.567 ± 1.354	0.783
5	DF	66.333 ± 3.0578	94.533 ± 2.050	0.126
	PCA	84.856 ± 3.544	88.733 ± 0.814	0.825

Note that in cases where more than 30 features were used, the performance of the ensemble declined with the addition of additional features, as more and more irrelevant features were taken into account. Indeed, for 30 or fewer features, input decimation significantly outperformed PCA while for 40 or more features, input decimation only had marginally higher performance. However, except for the 70-feature ensemble, all the input decimation ensembles provided statistically significant improvements over the original ensembles. Also, note that the single decimated networks with 20 and more features outperformed the original single classifier. This perhaps surprising result (as one might have expected only the ensemble performance to improve with feature subsets) is mainly due to the simplification of the learning tasks, which allows the classifiers to learn the mapping more efficiently.

Interestingly, the correlation among classifiers does not decrease until a very small number of features remain. We attribute this to the removal of noise, which increases the amount of information shared between the classifiers. Indeed, the correlation increases steadily as

Table 3: Synthetic Dataset 2: Influence of Dimensionality on Ensemble Performances

Dim.		Single	Ensemble	Corr.
70	DF	84.767 ± 2.419	90.000 ± 1.955	0.717
	PCA	84.422 ± 2.625	89.600 ± 1.902	0.729
60	DF	85.778 ± 3.197	91.533 ± 1.968	0.733
	PCA	85.922 ± 2.724	90.533 ± 1.681	0.742
50	DF	87.611 ± 2.532	92.233 ± 1.567	0.761
	PCA	86.767 ± 2.370	91.033 ± 1.949	0.773
40	DF	89.667 ± 2.193	93.700 ± 1.043	0.823
	PCA	79.567 ± 2.416	88.333 ± 2.071	0.659
30	DF	90.067 ± 2.508	94.500 ± 1.364	0.814
	PCA	80.078 ± 2.502	90.667 ± 1.862	0.675
20	DF	87.089 ± 2.094	95.467 ± 1.343	0.638
	PCA	80.611 ± 2.353	90.267 ± 1.781	0.690
10	DF	67.356 ± 2.601	93.400 ± 2.054	0.153
	PCA	80.111 ± 2.006	89.600 ± 1.890	0.714
5	DF	66.100 ± 3.038	90.733 ± 2.520	0.145
	PCA	78.678 ± 2.057	88.333 ± 1.291	0.743

features are removed until we reach 30 features (which corresponds to the actual number of relevant features). After that point, removing features reduces the correlation and the individual classfier performance. However, the ensemble performance still remains high. This experiment clearly shows the trade-off presented in Equation 4: one can either increase individual classifier performance (as for DF with more than 30 features) or reduce the correlation among classifiers (as for DF with less than 20 features) to improve ensemble performance.

4.1.2 Set 2

Table 3 presents the results for the second data set which is obtained by reducing the number of irrelevant features (from 70 to 50) from the first dataset.[5] The decimated ensembles with 20, 30, and 40 features outperformed the original ensemble and PCA-based ensemble significantly, while the 10-feature ensemble performed marginally better. The remaining decimated ensembles provided results that were statistically similar to those of the original ensemble.

[5]The single classifier used was an MLP with a single hidden layer consisting of 65 units, trained using a learning rate of 0.2 and a momentum term of 0.5.

Note that just as it was for the first data set, in this case, the single classifiers with 20 or more features outperformed the single original classifier, demonstrating the improvement we can achieve through dimensionality reduction alone, if the original feature set is noisy.

4.1.3 Set 3

Table 4 presents the results for the third data set, which is obtained by reducing the number of irrelevant features (from 70 to 20) from the first dataset.[6] That the original single classifier and ensemble perform better for this dataset relative to dataset 1 (see Table 1) is not surprising, because with fewer irrelevant features, there is less noise to "overfit." Therefore in this dataset, the gains due to input decimation are smaller. Indeed only the 10-dimensional decimated ensemble significantly outperformed the original ensemble while the others provided only marginal improvements.

Table 4: Synthetic Dataset 3: Influence of Dimensionality on Ensemble Performances

Dim.		Single	Ensemble	Corr.
40	DF	86.478 ± 2.389	91.633 ± 2.060	0.747
	PCA	87.222 ± 2.427	92.167 ± 1.412	0.760
30	DF	87.400 ± 2.826	92.333 ± 1.693	0.759
	PCA	88.367 ± 2.370	92.200 ± 1.621	0.790
20	DF	84.133 ± 2.461	90.933 ± 1.583	0.660
	PCA	89.411 ± 2.016	93.000 ± 1.498	0.834
10	DF	68.878 ± 2.810	94.167 ± 2.804	0.204
	PCA	91.056 ± 1.909	93.633 ± 0.977	0.870
5	DF	65.889 ± 3.045	90.933 ± 2.255	0.123
	PCA	92.211 ± 1.195	93.700 ± 1.064	0.894

4.1.4 Set 4

Table 5 presents the results for the fourth data set, which is obtained from the first dataset by increasing the number of examples in the training and test sets by tenfold. [7] The performance improvements due to decimation are smaller here than they were for the previous datasets;

[6]The single classifier used was an MLP with a single hidden layer consisting of 45 units, trained using a learning rate of 0.2 and a momentum term of 0.5.

[7]The single classifier used was an MLP with a single hidden layer consisting of 95 units, trained using a learning rate of 0.2 and a momentum term of 0.5.

however, all the decimated ensembles with 20 or more features still significantly outperformed the original ensemble. In this case, single decimated classifiers with 20 or more features do not outperform the original classifiers to the same extent as they did for dataset 1. This is because with the increase in the number of samples, the original classifier has a better chance to extract the "signal" from the "noise" and thus is less affected by the irrelevant features.

Also, in this experiment the PCA based ensembles performed well and were only beaten by input decimated ensembles for subsets of 20 and 30 features. Furthermore, the first few principal components found by PCA carry good discriminating information in this case, explaining why there is such little variability between the performance of the PCA ensembles with varying numbers of features. Although the behavior of the correlation is very similar to that observed for Set 1, the actual correlation values are higher across the board. This is not surprising since with more data, the similarities between the classifiers are amplified.

Table 5: Synthetic Dataset 4: Influence of Dimensionality on Ensemble Performances

Dim.		Single	Ensemble	Corr.
70	DF	91.732 ± 0.614	94.107 ± 0.357	0.847
	PCA	92.078 ± 0.668	94.267 ± 0.125	0.851
60	DF	92.257 ± 0.565	94.433 ± 0.414	0.853
	PCA	92.213 ± 0.601	94.440 ± 0.480	0.854
50	DF	92.820 ± 0.513	94.780 ± 0.326	0.872
	PCA	93.078 ± 0.488	94.660 ± 0.477	0.869
40	DF	93.356 ± 0.634	95.040 ± 0.438	0.885
	PCA	93.299 ± 0.479	94.830 ± 0.299	0.880
30	DF	94.153 ± 0.516	95.683 ± 0.381	0.903
	PCA	93.581 ± 0.366	94.886 ± 0.328	0.893
20	DF	91.482 ± 0.895	97.380 ± 0.372	0.786
	PCA	93.968 ± 0.519	95.039 ± 0.416	0.905
10	DF	66.587 ± 0.660	93.113 ± 2.998	0.130
	PCA	94.408 ± 0.429	95.113 ± 0.298	0.924
5	DF	65.298 ± 4.806	89.463 ± 6.453	0.107
	PCA	94.520 ± 0.403	95.007 ± 0.288	0.942

4.1.5 Set 5

Table 6 presents the results for the fifth data set, which is similar to the first dataset but there is overlap among the relevant features for the classes. [8] Because of this overlap, this feature set has fewer total relevant features and thus it constitutes a more difficult problem (as indicated by the results in Table 1). This is also demonstrated by the similarity among the correlations for all the different subset sizes. Unlike with the previous four data sets, the correlation does not go down drastically here for a small subset, because the overlap among the classes forces the classifiers to remain "coupled" to one another.

Table 6: Synthetic Dataset 5: Influence of Dimensionality on Ensemble Performances

Dim.		Single	Ensemble	Corr.
70	DF	81.778 ± 2.792	87.567 ± 2.331	0.720
	PCA	79.822 ± 2.733	86.100 ± 2.173	0.706
60	DF	83.811 ± 2.704	89.333 ± 2.404	0.749
	PCA	80.422 ± 2.689	85.567 ± 2.036	0.735
50	DF	85.056 ± 2.605	90.233 ± 1.664	0.796
	PCA	81.056 ± 2.406	86.467 ± 1.335	0.729
40	DF	86.333 ± 2.433	91.100 ± 2.122	0.802
	PCA	79.933 ± 2.685	84.933 ± 1.389	0.732
30	DF	86.844 ± 2.155	91.467 ± 1.771	0.795
	PCA	79.878 ± 2.625	85.600 ± 1.254	0.732
20	DF	86.967 ± 2.632	92.267 ± 1.806	0.783
	PCA	79.656 ± 2.798	84.500 ± 1.590	0.743
10	DF	85.756 ± 2.825	98.133 ± 0.980	0.707
	PCA	79.122 ± 2.249	85.133 ± 1.910	0.755
5	DF	81.956 ± 4.192	95.467 ± 1.614	0.706
	PCA	70.856 ± 2.427	78.200 ± 1.507	0.683

In spite of these difficulties, input decimation ensembles perform extremely well. Indeed, they significantly outperform both the original ensemble and PCA ensembles on all but a few subsets where they only provide marginal improvements. Furthermore the input decimated single classifiers also outperform their original and PCA counterparts for all but the 60 and 70 feature subsets. This experiment demonstrates that when there is overlap among

[8] The single classifier used was an MLP with a single hidden layer consisting of 95 units, trained using a learning rate of 0.2 and a momentum term of 0.5.

classes, class information is crucial. Without this vital information, PCA cannot provide any statistically significant improvements over the original classifier and ensembles.

4.2 UCI/Proben1 Datasets

To complement the experiments discussed above, we also selected three datasets from the UCI/PROBEN1 benchmarks [8, 51]: The Gene database from the PROBEN1 (i.e., using train/validate/test split from PROBEN1), and the Splice junction gene sequences and Satellite Image database (Statlog version) from the UCI Machine Learning Repository. In these experiments, just as in those described above, our classifiers consist of MLPs.

4.2.1 Data Description and Full Feature Set Performance

In this section we provide a brief description of the data sets and the individual classifiers. The Gene dataset has 120 input features and three class variables [45, 51]. We selected a component MLPs with a single hidden layer of 20 units, a learning rate of 0.2 and a momentum term of 0.8. The Splice data consists of 60 input features and three classes [8]. 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 Image data has 36 input features and 6 classes [8]. We selected an MLP with a single hidden layer of 50 units, and a learning rate and momentum term of 0.5.

Table 7: Average Accuracy of Original Network and Ensembles

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

Table 7 provides the classification performance for single classifiers and ensembles on the full feature set for all three datasets[9]. For the Gene data, the average ensemble was significantly more accurate than the single network, while for the Satellite Image and Splice data sets, the ensemble was only marginally more accurate.

[9]The ensemble consists of 3 classifiers for the Gene and Splice datasets and of 6 classifiers for the Satellite Image dataset.

4.2.2 Fixed Input Decimated Ensembles

This section describes experiments that mirror those above where we investigate the performance of single classifiers and ensembles with "fixed" subsets of the features set (i.e., each classifier sees the *same* number of features). For the Gene and Splice datasets, we use increments of 10 features up to the full set, while for the Satellite Image data we use increments of 9 features. The classification performance for both the single classifiers and the ensembles on all subsets, averaged over 20 runs, along with the corresponding correlation values (i.e., correlation among classifiers in the ensemble) are given in Tables 8-10 below.

Table 8: Gene Data: Influence of Dimensionality on Ensemble Performances

Dim.		Single	Ensemble	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

In case of the Gene data, the average ensembles with 20, 30, and 40 inputs are significantly more accurate than both the original network ensembles described in the previous section and their PCA counterparts. Note also that the performances of the PCA-based ensembles vary arbitrarily as the number of principal components changes, while the performances of the feature-based ensembles 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 ensembles significantly outperformed both the original ensemble and the PCA-based ensembles. What is particularly

Table 9: Satellite Image Data: Influence of Dimensionality on Ensemble Performances

Dim.		Single	Ensemble	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 10: Splice Data: Influence of Dimensionality on Ensemble Performances

Dim.		Single	Ensemble	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

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 performance of the single classifiers drops by 7 %, whereas the performance of the DF single classifier increases by 3 %. The improvement of the performance of the single classifiers due to decimation is an initially surprising aspect of these experiments (unlike the synthetic data sets, one does not expect to find too many "irrelevant" features in these real datasets). However, an analysis shows that the inputs that were decimated were in fact providing "noise" to the classifier. Although it is theoretically true that the classifier with more information will do at least as well as the classifier with less information, in practice with only a limited amount of data, extracting the correct information can cause a problem for such classifiers causing them to perform worse than their counterparts with less information.

In the Satellite Image data however, the input decimated ensemble with 27 features was the only one that did not perform significantly worse than the single neural network and

the original ensemble. 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 provide good discriminative features. A potential solution to this problem is to select "wild card" features on correlation to the overall problem and include them in each decimated subset.

4.2.3 Variable Input Decimated Ensembles

With the UCI/Proben1 datasets there is no reason to assume that each of the classifiers in an ensemble should have the same number of features. Therefore we have performed experiments where we allowed the subsets to vary in size. To select the number of features for each class, we first plotted the correlation between each feature and one output class in decreasing order. We then selected the subset with the most natural break point as the significant features. The experiments reported below show the potential of using variable numbers of features. We are currently investigating more formal methods to automate the selection of the number of features for each classfier.

Table 11: Variable Input Ensembles.

Dataset	Features/Class	Single	Ensemble	Corr.
gene	11-8-14	82.211 ± 0.857	90.757 ± 0.615	0.6334
satellite	27-27-9-18-27	80.483 ± 0.890	88.370 ± 0.005	0.6361
splice	13-10-21	87.833 ± 0.641	92.371 ± 0.3351	0.7719

Table 11 provides the classification performance for single classifiers and ensembles on the decimated feature sets for the three data sets. The second column provides the number of features present in each of the classifiers (i.e., for Gene the first classifier in the ensemble had 11 features, the second had 8 while the third had 14). For the Gene database, the variable input decimated ensembles improved upon the fixed subset input decimation results (which themselves were an improvement over the original ensemble). For the splice dataset, the improvements over the original ensemble are even more drastic, although the results are

statistically equivalent to those obtained with fixed subsets of 10 and 20 features. As for the satellite image dataset, variable input decimated ensembles improved upon the fixed input decimated ensembles, but still fell short of the original ensembles (for the same reasons that we highlighted in Section 4.2.2).

5 Discussion and Conclusions

This paper discusses input decimation, a dimensionality reduction method for ensemble classification. We present experimental results demonstrating that input decimated ensembles are a promising machine learning method that yield superior results by combining the strengths of dimensionality reduction and ensembles. Specifically, we show that, in most cases, the single decimated classifiers outperform the single original classifier (trained on the full feature set), which demonstrates that simply eliminating irrelevant features improves performance. In addition, eliminating irrelevant features in each of many classifiers using *different relevance criteria* (in this case, relevance with respect to different classes) yields significant improvement in ensemble performance, as seen by comparing our decimated ensembles to the original ensembles. Selecting the features using class label information also provided significant performance gains over PCA-based ensembles. Furthermore, using subsets of the original features instead of new features allows human operators to gain more insight into how each classifier and ensemble makes its decisions, alleviating a serious difficulty in interpreting results in large data mining problems [20, 49].

Through our tests on real and synthetic datasets, we show certain characterizations that datasets need to have to fully benefit from input decimation. Namely, we show that input decimation performs best when there are a large number of features (e.g., where it's likely that there will be irrelevant features) and when the number of training examples is relatively small (e.g., where it's difficult to properly learn all the parameters in a classifier based on the full feature set). In these cases, decimation removes the extraneous features, thereby reducing noise and reducing the number of training examples needed to produce a meaningful model (i.e., alleviate the curse of dimensionality).

An interesting observation is that input decimation works well *in spite* of our rather crude method of choosing the relevant features (i.e., statistical correlation). One reason why this simply method succeeds is that we have greatly simplified the relevance criterion: only the relevance of the features to *a single* output is taken into consideration, rather than focus on the discriminatory ability across all classes. Nevertheless, we are currently extending this work in three directions: considering cross-correlations among the features; investigating mutual

information based relevance criteria; and incorporating global relevance into the selection process. We are confident that a fully developed input decimated ensemble method will provide an easy to use, understandable and robust method for addressing high-dimensional classification problems that are common in data mining.

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

References

[1] K. Al-Ghoneim and B. V. K. Vijaya Kumar. Learning ranks with neural networks (Invited paper). In *Applications and Science of Artificial Neural Networks, Proceedings of the SPIE*, volume 2492, pages 446–464, April 1995.

[2] 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.

[3] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5:4:537–550, July 1994.

[4] R. Battiti and A. M. Colla. Democracy in neural nets: Voting schemes for classification. *Neural Networks*, 7(4):691–709, 1994.

[5] S. D. Bay. Combining nearest neighbor classifiers through multiple feature subsets. In *Proceedings of the 15th International Conference on Machine Learning*, pages 415–425. Morgan Kaufmann, 1998.

[6] J.A. Benediktsson, J.R. Sveinsson, O.K. Ersoy, and P.H. Swain. Parallel consensual neural networks with optimally weighted outputs. In *Proceedings of the World Congress on Neural Networks*, pages III:129–137. INNS Press, 1994.

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

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

[9] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–272, 1997.

[10] K. D. Bollacker and J. Ghosh. Linear feature extractors based on mutual information. In *Proc. of the 13th Intl. Conf. on Pattern Recognition*, pages pp. IV:720–724, 1996.

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

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

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

[14] C. E. Brodley and P. Smyth. Applying classification algorithms in practice. *Statistics and Computing*, 7:45–56, 1997.

[15] 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.

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

[17] T.G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1998.

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

[19] P. Domingos. Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review*, 11:227–253, 1997.

[20] P. Domingos. Knowledge discovery via multiple models. *Intelligent Data Analysis*, 2:187–202, 1998.

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

[22] H. Drucker, R. Schapire, and P. Simard. Improving performance in neural networks using a boosting algorithm. In S.J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems-5*, pages 42–49. Morgan Kaufmann, 1993.

[23] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI Press/MIT Press, 1996.

[24] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc. of the 13th Intl. Conf. on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[25] C. Glymour, D. Madigan, D. Pregibon, and P. Smyth. Statistical themes and lessons for data mining. *Data Mining and Knowledge Discovery*, 1(1):11–28, 1997.

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

[27] S. Hashem and B. Schmeiser. Approximating a function and its derivatives using MSE-optimal linear combinations of trained feedforward neural networks. In *Proceedings of the Joint Conference on Neural Networks*, volume 87, pages I:617–620, New Jersey, 1993.

[28] T. Heskes. Balancing between bagging and bumping. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems-9*, pages 466–472. M.I.T. Press, 1997.

[29] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Tran. on Pattern Analysis and Machine Intelligence*, 16(1):66–76, 1994.

[30] A. Hyvarinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

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

[32] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the International Conference on Machine Learning*, pages 121–129, July 1994.

[33] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[34] M. I. Jordan and R. A. Jacobs. Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[35] 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.

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

[37] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence Journal*, 1-2:273–324, 1997.

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

[39] S. Kumar, M. Crawford, and J. Ghosh. A versatile framework for labelling imagery with a large number of classes. In *Proceedings of the International Joint Conference on Neural Networks*, 1999.

[40] P. Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance*, 1994.

[41] 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.

[42] W.P. Lincoln and J. Skrzypek. Synergy of clustering multiple back propagation networks. In D. Touretzky, editor, *Advances in Neural Information Processing Systems-2*, pages 650–657. Morgan Kaufmann, 1990.

[43] 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.

[44] C. J. Merz. A principal component approach to combining regression estimates. *Machine Learning*, 36:9–32, 1999.

[45] 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, *Advances in Neural Information Processing Systems-3*, pages 530–536. Morgan Kaufmann, 1991.

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

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

[48] M. Partridge and R. A. Calvo. Fast dimensionality reduction and simple pca. *Intelligent Data Analysis*, 2:203–214, 1998.

[49] M. Pazzani. Comprehensible knowledge discovery: gaining insight from data. In *First Federal Data Mining Conference and Exposition*, pages 73–82, Washington, DC, 1997.

[50] M.P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Neural Networks for Speech and Image Processing*, chapter 10. Chapmann-Hall, 1993.

[51] 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.

[52] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufman, San Mateo, California, 1992.

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

[54] G. Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7(5):777–781, 1994.

[55] 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.

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

[57] A. J. J. Sharkey. (editor). *Connection Science: Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3 & 4), 1996.

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

[59] 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.

[60] K. Tumer and J. Ghosh. Classifier combining through trimmed means and order statistics. In *Proceedings of the International Joint Conference on Neural Networks*, pages 757–762, Anchorage, Alaska, 1998.

[61] K. Tumer and N. C. Oza. Decimated input ensembles for improved generalization. In *Proceedings of the International Joint Conference on Neural Networks*, 1999.

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

[63] N. Wyse, R. Dubes, and A.K. Jain. A critical evaluation of intrinsic dimensionality algorithms. In E. Gelsema and L.N. Kanal, editors, *Pattern Recognition in Practice2*, pages 415–425. Morgan Kaufmann, 1980.

[64] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418–435, May 1992.

[65] J.-B. Yang and M. G. Singh. An evidential reasoning approach for multiple-attribute decision making with uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(1):1–19, 1994.