Input Decimation Ensembles: Decorrelation through Dimensionality Reduction

Nikunj C. Oza[1] and Kagan Tumer[2]

[1] Computer Science Division
University of California
Berkeley, CA 94720-1776, USA
oza@cs.berkeley.edu
[2] Computational Sciences Division
NASA Ames Research Center
Mail Stop 269-3
Moffett Field, CA 94035-1000, USA
kagan@ptolemy.arc.nasa.gov

Abstract. Using an ensemble of classifiers instead of a single classifier has been shown to improve generalization performance in many machine learning problems [4, 16]. However, the extent of such improvement depends greatly on the amount of correlation among the errors of the base classifiers [1, 14]. As such, reducing those correlations while keeping the base classifiers' performance levels high is a promising research topic. In this paper, we describe *input decimation*, a method that decouples the base classifiers by training them with different subsets of the input features. In past work [15], we showed the theoretical benefits of input decimation and presented its application to a handful of real data sets. In this paper, we provide a systematic study of input decimation on synthetic data sets and analyze how the interaction between correlation and performance in base classifiers affects ensemble performance.

1 Introduction

Using an ensemble of classifiers instead of a single classifier has been repeatedly shown to improve generalization performance in many machine learning problems [4, 16]. It is well-known that, in order to obtain such improvement, one needs to simultaneously maintain a reasonable level of performance in the base classifiers that constitute the ensemble and reduce their correlations. There are many ensemble methods that actively promote diversity (i.e., lower correlations in the outputs) among their base classifiers. Bagging [4], boosting [7], and cross-validation partitioning [9, 14] generate diverse base classifiers by training with different subsets of the training set. Error-correcting output codes [5] generate new training sets with different class labels and use these different training sets to generate base classifiers. Merz [10] use Principal Component Analysis [8] to measure the correlations among the base models and combine them accordingly. Dietterich [6] combines decision trees in which each test is chosen at random among the 20 best tests.

Most work in this field, however, focuses on pattern-level selection (e.g., Bagging, Boosting). **Input Decimation (ID)** on the other hand is a feature selection method that generates different subsets of the input features for each of the classifiers in the ensemble. By training each base classifier with a different feature subset, the correlations among the base classifiers are reduced. (Note that input decimation can be used in conjunction with pattern-based ensemble methods such as bagging and boosting, as discussed in Section 3.) Input decimation is different from most other dimensionality reduction methods that are widely used, including PCA, in that it generates different feature subsets for different classifiers. On the other hand, PCA aims to maximize the variability among the newly constructed features, but makes no provisions on how that variability is related to class information (see [11] for details).

In this work we explore using class information to reduce the dimensionality of the feature space presented to each base classifier. While strong ensemble performance was expected, input decimation also provided improvements in the base classifiers by pruning *irrelevant* features, thereby simplifying the learning problem faced by each base classifier. Consequently, Input Decimated Ensembles (IDEs) significantly outperformed both base classifiers trained on the full feature space as well as ensembles of such classifiers. In the next section we briefly highlight the need for correlation reduction in ensembles. We then present the input decimation algorithm, along with results on synthetic data sets.

2 Correlation and Ensemble Performance

In this article we focus on classifiers that model the *a posteriori* probabilities of the output classes. Such algorithms include Bayesian methods [3], and properly trained feed forward neural networks such as Multi-Layer Perceptrons (MLPs) [12]. We can model the ith output of such a classifier as follows (details of this derivation are in [13, 14]):

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

where $P(C_i|x)$ is the posterior probability 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 an ensemble that calculates the arithmetic average over the outputs of N classifiers $f_i^m(x)$, $m \in \{1, \ldots, N\}$, then $P(C_i|x)$ is given by:

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

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 [14]:

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

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

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

then the variance becomes:

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

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 model error (beyond the Bayes error) of the ensemble (E^{ave}_{model})and that of an individual classifier (E^{ave}_{model}) [13, 14]:

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

where

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

and 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 base 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 concept which merges dimensionality reduction and correlation reduction to provide classifier ensembles.

3 The Input Decimated Ensembles

Input decimation decouples the classifiers by exposing them to different aspects of the same data by selecting features most correlated with a particular class. ID trains L classifiers, one corresponding to each class in an L-class problem[1]. For each classifier, the method selects a user-determined number of the input

[1] More generally, one trains nL classifiers where n is an integer.

features having the highest absolute correlation to the presence or absence of the corresponding class[2]. The objective is to "weed" out input features that do not carry strong discriminating information for a particular class, and thereby reduce the dimensionality of the feature space to facilitate the learning process.

Let the training set take the following form:

$$\{(\mathbf{x_1}, \mathbf{y_1}), (\mathbf{x_2}, \mathbf{y_2}), \ldots, (\mathbf{x_m}, \mathbf{y_m})\},$$

where m is the number of training examples. Each $\mathbf{x_i}$ has $\|FS\|$ elements (where FS is the set of input features) representing the values of the input features in example i. Each $\mathbf{y_i}$ represents the class using a distributed encoding, i.e., it has L elements, where L is the number of classes, $y_{il} = 1$ if example i is an instance of class l and $y_{il} = 0$ if example i is not an instance of class l. In this study our base classifiers consist of MLPs trained with the backpropagation algorithm[3].

Given such a data set, and a base classifier learning algorithm, input decimated ensembles operate as follows:

- For each class $l \in \{1, 2, \ldots, L\}$,
  1. Compute the absolute value of the correlation between each feature j ($\mathbf{x_{ij}}$ for all patterns i) and the output for class l ($\mathbf{y_{il}}$ for all patterns i).
  2. Select the n_l features having the highest absolute correlation, resulting in new feature set FS_l. One can either predetermine n_l based on prior information about the data set, or learn the value to optimizes performance.
  3. Construct a new training set by retaining only those elements of the $\mathbf{x_i}$'s corresponding to the features FS_l and all the outputs.
  4. Call the base classifier learning algorithm on this new training set. Call the resulting classifier f^l.

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

- For each class $k \in \{1, 2, \ldots, L\}$, calculate $f_k^{ave}(x) = \frac{1}{L}\sum_{l=1}^{L} f_k^l(x)$, by presenting the proper feature sets (FS_l) to each of the L classifiers.
- Return the class $K = argmax_k f_k^{ave}(x)$.

Fundamentally, input decimation seeks to reduce the correlations among individual classifiers by using different subsets of input features, while methods such as bagging and boosting attempt to do so by choosing different subsets of training 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 pattern-based methods, and directly comparing

[2] Note that this method requires the problem to have at least three classes. In a two-class problem, features strongly correlated with one class will be strongly anti-correlated with the other class, so the same features would be chosen for both classifiers.

[3] In principle, any learning algorithm that estimates the a posteriori class probabilities can be used.

input decimation to bagging or boosting serves little purpose. Rather one should compare input decimated ensembles to original ensembles (which is done here) or input decimated, *bagged* ensembles to bagging alone (which we are currently investigating).

4 Experimental Results

In this section, we present the results of input decimation on synthetic datasets. As discussed above, our base classifiers are multi-layer perceptrons. In this work all such classifiers contain a single hidden layer and the learning rate, momentum term, and number of hidden units were experimentally determined[4].

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 "original single classifier") and separately trained L copies of the same classifier and incorporated them into an averaging ensemble (referred to as the "original ensemble"). As anticipated, the original ensemble often performs significantly better than each of its base classifiers. Comparing input-decimated ensembles with these original ensembles isolates the benefits of removing input features from the base classifiers. Because PCA is a standard dimensionality reduction method, we also compare input decimated ensembles to PCA ensembles (i.e., ensembles where each constituent classifier was trained on a preselected set of the principal components of the feature space).

In these experiments, we used the following three synthetic datasets:

- Set 1:
  - Three classes–one unimodal Gaussian per class.
  - 300 training patterns and 150 test patterns–100 training and 50 test patterns per class.
  - 100 features per pattern where there are:
    - 10 relevant features per class–each class's instances are generated from a multivariate normal distribution in 10 independent dimensions distributed as $N(40, 5^2)$. There are no dimensions in common among the three classes. Therefore, there are 30 relevant features. For instances of each class, the 20 features that are relevant to the other two classes are distributed as $U[-100, 100]$.[5]
    - 70 irrelevant features–distributed as $U[-100, 100]$.
- Set 2: Same as Set 1, except that only 50 irrelevant features were added to the 30 relevant features, for a total of 80 features in the dataset.
- Set 3: Same as Set 1, except that there is overlap among the relevant features for each class (e.g., classes have three relevant features in common).

[4] We experimented on a single neural network with all input features by trying learning rates and momentum terms in increments of 0.05 and hidden units in increments of 5 until the performance began to decline.

[5] Clearly, because of this, all 30 features have some relevance to all three classes; however, the 10 features used to generate each class's instances are clearly substantially more relevant than the other 20 features.

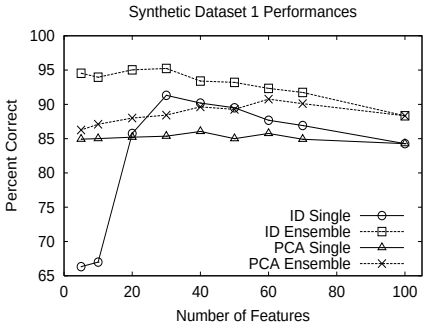

Fig. 1. Dataset 1 Performances

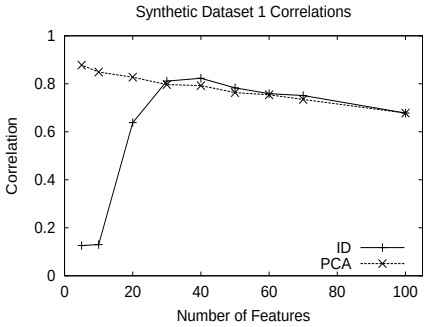

Fig. 2. Dataset 1 Correlations

In dataset 1 there is an abundance of features that are irrelevant for the classification task. This data set was chosen to represent large data mining problems where the algorithms may get swamped by irrelevant data. Dataset 2 has fewer irrelevant features and was chosen to illustrate the performance of input decimation as a function of irrelevant information present in the feature space. By reducing the amount of noise in the feature space, the problem is subtly modified: selecting the relevant features is now easier, but the effect of removing the irrelevant features on the base classifiers' performance is reduced. Finally, dataset 3 was chosen to have overlap among the features relevant to each class. This provides a more difficult problem where the base classifiers are now *forced* to select some common features, reducing the potential for correlation reduction.

4.1 Synthetic Set 1

Figures 1 and 2 present the classification accuracies and base classifier correlations, respectively as a function of the number of inputs (which are either the number of selected principal components or the number of features selected for each base classifier through input decimation). The original single classifier and original ensemble use all the input features[6]. The points for the maximum number of features (e.g., 100 features in this dataset), always represent the performance of the original classifier/ensemble.

An important observation that is apparent from these results is that neither PCA ensembles nor PCA base classifiers are particularly sensitive to the number of inputs. The correlations among the base classifiers reinforce this conclusion. Fewer input features in PCA means the base classifiers are more correlated since they all share the same principal features. Note however, that input decimated base classifiers have little correlation for small numbers of features, increasing correlation up to 30 features, and decreasing correlation after that. The base classifiers' average performance follows a similar pattern. Interestingly though,

[6] The base 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.

input decimated ensembles are not adversely affected by the poor performance of the base classifiers (e.g., input decimated ensembles with 5 features outperformed input decimated ensembles with 50 features while base classifiers with 5 features gave significantly worse results than base classifiers with 50 features).

In cases where more than 30 features were used, the performance of the ensemble declined with the addition of additional features, i.e., as more and more irrelevant features were included. However, all the input decimation ensembles provided statistically significant improvements over the original ensembles and PCA ensembles.

The single decimated classifiers 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 when using subsets of the features) is mainly due to the simplification of the learning tasks, which allows the classifiers to learn the mapping more efficiently.

Interestingly, the average correlation among classifiers does not decrease until a very small number of features remain. We attribute this to the removal of noise—removing noise increases the amount of information shared between the base classifiers. Indeed, the correlation increases steadily as 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 classifier performances. However, the ensemble performance still remains high. This experiment clearly shows a typical trade-off in ensemble learning: one can either increase individual classifier performance (as for input decimation with more than 30 features) or reduce the correlation among classifiers (as for input decimation with less than 20 features) to improve ensemble performance.

4.2 Synthetic Set 2

Figures 3 and 4 present the classification accuracies and base classifier correlations, respectively, for the second data set which is obtained by reducing the number of irrelevant features (from 70 to 50) from the first dataset[7]. The decimated ensembles with 5 and 70 features marginally outperformed the original ensemble and PCA-based ensemble, while the remaining ones performed significantly better. Note that, just as it was for the first data set, the input decimated single classifiers with 20 or more features outperformed the single original classifier. This demonstrates that if the feature set is noisy (an assumption that almost always holds in the real world) improvements are achieved through dimensionality reduction alone.

4.3 Synthetic Set 3

Figures 5 and 6 present the results for the third data set, which is similar to the first dataset except that there is overlap among the relevant features for the

[7] 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.

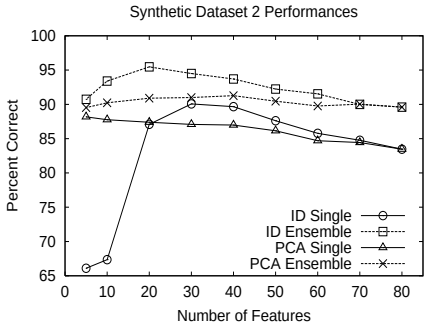

Fig. 3. Dataset 2 Performances

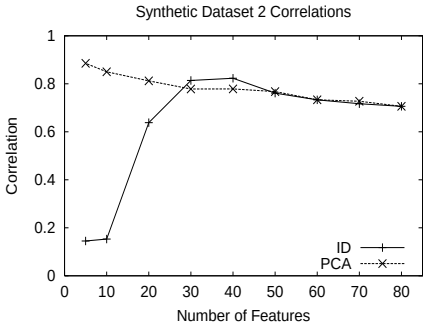

Fig. 4. Dataset 2 Correlations

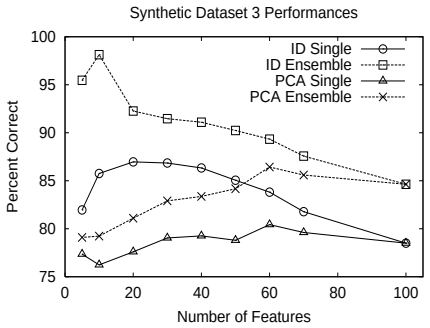

Fig. 5. Dataset 3 Performances

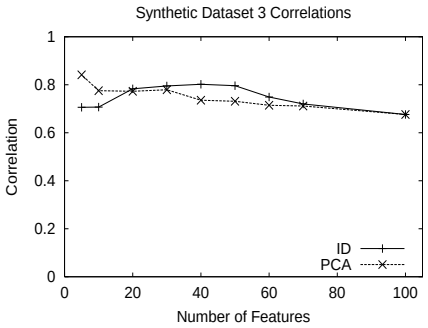

Fig. 6. Dataset 3 Correlations

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 comparing the results on the full feature classifiers and ensembles on this dataset to the previous ones).

Note that the correlations in this data set remained fairly constant across the board. Unlike results shown in Figure 2, input decimation did not reduce correlations dramatically for small feature sets. This is mainly caused by the "coupling" among the features (i.e., the presence of features that are essential to many classes due to the overlap).

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 is particularly heartening since this feature set is a more representative abstraction

[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.

of real data sets (data sets with "clean" separation among classes are quite rare). This experiment demonstrates that when there is overlap among classes, class information becomes particularly relevant. PCA operates without this vital information, therefore it cannot provide any statistically significant improvements over the original classifiers and ensembles.

5 Discussion

This paper discusses input decimation, a dimensionality reduction-based ensemble method that provides good generalization by reducing the correlations among the classifiers in the ensemble. Through controlled experiments, we show that the input decimated single classifiers outperform the single original classifiers (trained on the full feature set), demonstrating that simply eliminating irrelevant features can improve performance[9]. 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 provides significant performance gains over PCA-based ensembles.[10]

Through our tests on synthetic datasets, we examined the characteristics that datasets need to have to fully benefit from input decimation. We observed that input decimation performs best when (i) there are a large number of features (i.e., where it's likely that there will be irrelevant features); and (ii) when the number of training examples is relatively small (i.e., where it's difficult to properly learn all the parameters in a classifier based on the full feature set). In both cases, by removing the extraneous features, input decimation reduces noise and thereby reduces the number of training examples needed to produce a meaningful model (i.e., alleviating the curse of dimensionality). Our synthetic datasets were generated using multivariate distributions where the feature values were generated independently. We plan to generate synthetic datasets with dependencies among the features to see how they affect our method.

Note that input decimation shares the central aim of generating a diverse pool of classifiers for the ensemble with many methods, and most notably with bagging. However, by focusing on the input features rather than the input patterns, input decimation focuses on a different "axis" of correlation reduction than does bagging. Consequently, input decimation is orthogonal to bagging, and one can use input decimation in conjunction with bagging.

A final observation is that input decimation works well *in spite* of our rather crude method of feature selection (i.e., using statistical correlation of each feature individually with each class). One reason why this simple method succeeds

[9] Although this result is perplexing from an information theory perspective, it is consistent with learning theory: by removing features we simplify the learning task and thus allow the base classifiers to reach their "peak" performance.

[10] Furthermore, IDEs also outperform random feature subset selection [2, 17] on real datasets [15].

is that we have greatly simplified the relevance criterion: unlike other feature selection methods that consider the discriminatory ability across all classes, we only consider the relevance of the features to *a single* class. This typically causes each classifier in the ensemble to get a different subset of features, leading to the superior performance we have demonstrated. 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.

Acknowledgments. Part of this work was done while Nikunj Oza was visiting NASA Ames Research Center.

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. S. D. Bay. Combining nearest neighbor classifiers through multiple feature subsets. In *Proc. 15th ICML*, pages 415–425. Morgan Kaufmann, 1998.
3. J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. (2nd Ed.), Springer, New York, 1985.
4. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
5. T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of AI Research*, 2:263–286, 1995.
6. Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–158, Aug. 2000.
7. Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th ICML*, pages 148–156, Bari, Italy, 1996. Morgan Kaufmann.
8. I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.
9. 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.
10. C. J. Merz. *Classification and Regression by Combining Models*. PhD thesis, University of California, Irvine, Irvine, CA, May 1998.
11. N. C. Oza and K. Tumer. Dimensionality reduction through classifier ensembles. Technical Report NASA-ARC-IC-1999-126, NASA Ames Research Center, 1999.
12. M.D. Richard and R.P. Lippmann. Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Computation*, 3(4):461–483, 1991.
13. K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, February 1996.
14. 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.
15. K. Tumer and N. C. Oza. Decimated input ensembles for improved generalization. In *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN-99)*, 1999.
16. D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.
17. Z. Zheng and G.I. Webb. Stochastic attribute selection committees. In *Proc. of the 11th Australian Joint Conf. on AI (AI'98)*, pages 321–332, 1998.