# Class Decomposition via Clustering: A New Framework for Low-Variance Classifiers

Ricardo Vilalta, Murali-Krishna Achari, and Christoph F. Eick
Department of Computer Science
University of Houston
Houston TX, 77204-3010, USA
{vilalta, amkchari, ceick}@cs.uh.edu

## Abstract

*We propose a pre-processing step to classification that applies a clustering algorithm to the training set to discover local patterns in the attribute or input space. We demonstrate how this knowledge can be exploited to enhance the predictive accuracy of simple classifiers. Our focus is mainly on classifiers characterized by high bias but low variance (e.g., linear classifiers); these classifiers experience difficulty in delineating class boundaries over the input space when a class distributes in complex ways. Decomposing classes into clusters makes the new class distribution easier to approximate and provides a viable way to reduce bias while limiting the growth in variance. Experimental results on real-world domains show an advantage in predictive accuracy when clustering is used as a pre-processing step to classification.*

## 1 Introduction

Our study explores how classification algorithms can benefit from class density information that is obtained using clustering. Such information can be exploited to improve the quality of the decision boundaries during classification and enhance the prediction accuracy of simple classifiers. We demonstrate how using classification and clustering techniques in conjunction addresses key issues in learning theory (e.g., bias vs variance) and provides an attractive new family of classification models.

Our goal is to exploit the information derived from a clustering algorithm to increase the complexity of simple classifiers characterized by low variance and high bias. These algorithms, commonly referred to as model-based or parametric-based, encompass a small class of approximating functions and exhibit limited flexibility in their decision boundaries. Examples include linear classifiers, probabilistic classifiers based on the attribute-independence assumption (e.g., Naive Bayes), and single logical rules. The question we address is how to increase the complexity of these classifiers to trade-off bias for variance in an effective manner. Since these models start off with simple representations, increasing their complexity is expected to improve their generalization performance while still retaining their ability to output models amenable to interpretation.

Our approach extends previous work [6] in which we increase the degree of complexity of the decision boundaries of a simple classifier by augmenting the number of boundaries per class. The idea is to transform the classification problem by decomposing each class into clusters. By relabelling the examples covered by each cluster with a new class label, the simple classifier generates an increased number of boundaries per class, and is then armed to cope with complex distributions where classes cover different regions of the input space. In this paper we add a new step in which we explore the space of possible new class assignments in a greedy manner maximizing predictive accuracy.

We test our methodology on twenty datasets from the University of California at Irvine repository, using two simple classifiers: Naive Bayes and a Support Vector Machine with a polynomial kernel of degree one. Empirical results support our goal statement that *pre-identifying local patterns in the data through clustering is a helpful tool in improving the performance of simple classifiers*.

The paper organization is described next. Section 2 introduces background information and our problem statement. Section 3 details our class decomposition approach. Section 4 reports our empirical results. Finally, Section 5 states our summary and future work.

## 2 Problem Statement

Let $(X_1, X_2, \cdots, X_n)$ be an n-component vector-valued random variable, where each $X_i$ represents an attribute or feature; the space of all possible attribute vectors is called the attribute or input space $\mathcal{X}$. Let $\{y_1, y_2, \cdots, y_k\}$ be the possible classes, categories, or states of nature; the space of all possible classes is called the output space $\mathcal{Y}$. A classifier receives as input a set of training examples $T = \{(\mathbf{x}, y)\}$, where $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ is a vector or point in the input space ($x_i$ is the value of attribute $X_i$) and $y$ is a point in the output space. The outcome of the classifier is a function $h$ (or hypothesis) mapping the input space to the output space, $h : \mathcal{X} \rightarrow \mathcal{Y}$.

## 2.1 Simple Discriminant Functions

We consider the case where a classifier defines a discriminant function for each class $g_j(\mathbf{x})$, $j = 1, 2, \cdots, k$ and chooses the class corresponding to the discriminant function with highest value (ties are broken arbitrarily):

$$h(\mathbf{x}) = y_m \ \texttt{iff} \ g_m(\mathbf{x}) \geq g_j(\mathbf{x}) \qquad (1)$$

Possibly, the simplest case is that of a linear discriminant function, where the approximation is based on a linear model:

$$g_j(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i \qquad (2)$$

where each $w_i, 0 \leq i \leq n$, is a coefficient that must be learned by the classification algorithm.

We will also consider probabilistic classifiers where the discriminant functions are proportional to the posterior probabilities of a class given the input vector $\mathbf{x}$, $P(y_j|\mathbf{x})$. The classifier, also known as Naive Bayes, assumes feature independence given the class:

$$g_j(\mathbf{x}) = P(y_j)\Pi_i^n P(x_i|y_j) \qquad (3)$$

where $P(y_j)$ is the a priori probability of class $y_j$, and $\Pi_i^n P(x_i|y_j)$ is a simple product approximation of $P(\mathbf{x}|y_j)$, called the likelihood or class-conditional probability.

## 2.2 The Bias-Variance Trade-Off

Simple discriminant functions tend to output poor function approximations when the data distributes in complex ways. Our goal is to increase the complexity of simple classifiers to obtain better function approximations. Since our training set comprises a limited number of examples and we do not know the form of the true target distribution, our goal is inevitably subject to the bias-variance dilemma in statistical inference [4, 5]. The dilemma is based on the fact that prediction error can be decomposed into a bias and a variance component[1]; ideally we would like to have classifiers with low bias and low variance but these components are inversely related.

[1]A third component, the *irreducible error* or *Bayes error*, cannot be eliminated.

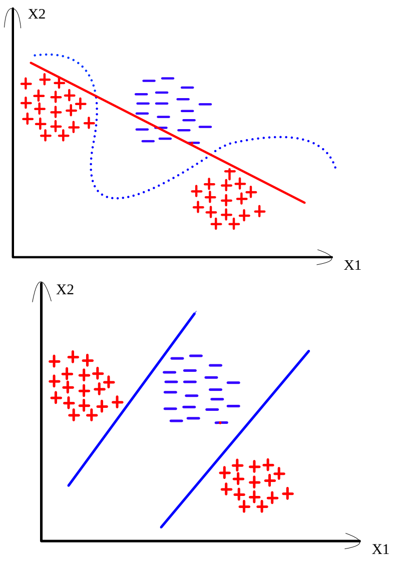

**Figure 1. (top) A high-order polynomial improves the classification of a linear classifier at the expense of increased variance. (bottom) Increasing the number of linear discriminants guided by local patterns increases complexity with lower impact on variance.**

Our problem statement can be rephrased as follows: how can we decrease the bias (i.e., increase the complexity) of our simple classifiers without drastically increasing the variance component? Notice our goal sets forth in a direction orthogonal to combination methods like bagging [2] and boosting [3] where the goal is to reduce the variance component in generalization error by voting on classifiers obtained from variants of the training data.

## 2.3 Increasing Complexity Through Additional Boundaries

Our solution is to exploit information about the distribution of examples through a pre-processing step that identifies natural clusters in data. As an illustration, Figure 1 shows a two dimensional input space with two classes (positive $+$ and negative $-$). The distribution of examples precludes a simple linear classifier attaining good performance (Figure 1-top, bold line). One way to increase the complexity of the classifier is to enlarge the original space of linear combinations to allow for more flexibility on the decision boundaries, for example by adding higher order polynomials (Figure 1-top, dashed line). But this comes at the expense of increased variance and possibly data overfitting.

Alternatively, one can retain the same space of linear functions but increase the number of decision boundaries per class (Figure 1-bottom). This increases the complexity of the classifier but with less impact on variance. The trick lies on identifying regions of high class density within subsets of examples of the same class which we accomplish through clustering. The next sections provide a detail description of our approach.

## 3 Class Decomposition via Clustering

Our solution comprises three steps: A) a decomposition of classes into clusters; B) a search for an optimal class assignment configuration; and C) a function mapping predictions to the original set of class labels. We explain each step in turn.

**A. Class Decomposition**. Our first step pre-processes the training data by clustering examples that belong to the same class. We proceed by first separating dataset $T$ into sets of examples of the same class. That is $T$ is separated into different sets of examples $T = \{T_j\}$, where each $T_j$ comprises all examples in $T$ labelled with class $y_j$, $T_j = \{(\mathbf{x}, y) \in T | y = y_j\}$.

For each set $T_j$ we apply a clustering algorithm $C$ to find sets of examples (i.e., clusters) grouped together according to some distance metric over the input space. Let $\{c_i^j\}$ be the set of such clusters. We map the set of examples in $T_j$ into a new set $T'_j$ by renaming every class label to indicate not only the class but also the cluster to which each example belongs. One simple way to do this is by making each class label a pair $(a, b)$, where the first element represents the original class and the second element represents the cluster that the example falls into. In that case, $T'_j = \{(\mathbf{x}, y'_j)\}$, where $y'_j = (y_j, i)$ whenever example $\mathbf{x}$ is assigned to cluster $c_i^j$. Finally the new dataset $T'$ is simply the union of all sets of examples of the same class relabelled according to the cluster to which each example belongs, $T' = \bigcup_{j=1}^{k} T'_j$.

An illustration of the transformation above is shown in Figure 2. We assume a two-dimensional input space where examples belong to either class positive $(+)$ or negative $(-)$. Let's suppose the clustering algorithm separates class positive into two clusters, while class negative is grouped into one single cluster. The transformation relabels every example to encode class and cluster label. As a result, dataset $T'$ has now three different classes.

**B. A Search for Class Assignments**. Increasing the number of classes according to the number of induced clusters may result in an excessive number of classes. Our second step extends previous work [6] by exploring the space of possible ways to merge clusters derived from the first step. Following the same notation as before, a class label will be a pair $(a, b)$, where the first element represents the original class label and the second element represents the cluster that the example falls into; but the difference now is that two or

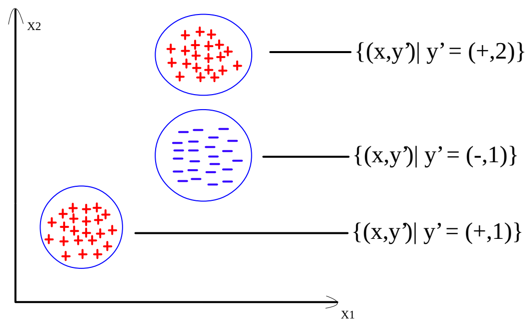

**Figure 2. The mapping process relabels examples to encode both class and cluster.**

more clusters may correspond to the same second element (i.e., element b), which can be interpreted as having clusters merged into a single cluster.

Our goal is to explore the space of possible ways to merge clusters obtained during the first step, until we find a configuration that maximizes predictive accuracy (over a validation set different from the training set). The space of possible configurations corresponds to the space of all subsets of clusters, with each subset being assigned the same cluster index (i.e., being assigned the same class label). Obviously one cannot explore this space exhaustively. If class $y_j$ is decomposed into $n_j$ clusters, the number of different configurations has an upper bound of $\mathcal{O}(2^{n_j})$. To avoid an exhaustive search we follow a heuristic greedy approach. The search starts by evaluating predictive accuracy assuming each cluster is mapped to a separate index. Next we start looking for pairs of clusters (e.g. $\{c_1^j, c_2^j\}$) and compute predictive accuracy assuming the two clusters of each pair are mapped to the same index. We then take those pairs for which predictive accuracy increased and rank them accordingly. To enforce a mutually exclusive list of clusters we prune every cluster pair where at least one cluster appears on another pair with higher rank. The algorithm keep merging clusters until no new cluster sets of higher cardinality can be produced from the cluster sets in the previous iteration. At that point we assign the clusters on each subset the same index (i.e., the same class label).

**C. Classification of Examples**. Our last step serves to assess the performance of the linear classifier over the extended output space. This is necessary during the search over the space of subsets of clusters, and while estimating final predictive accuracy. During learning, the simple classifier is trained over dataset $T'$ producing a hypothesis $h'$ mapping points from input space $\mathcal{X}$ to the new output space $\mathcal{Y}'$. During classification, hypothesis $h'$ will output a prediction consisting of a class label and a cluster label, $h'(\mathbf{x}) = (a, b)$. To know the actual prediction in the original output space $\mathcal{Y}$ we simply remove the cluster index.

Essentially, we predict class label $y_j$ whenever example $\mathbf{x}$ is assigned to any of the clusters (or subsets of clusters) of class $y_j$.

## 4 Experiments

Our experiments include as simple classifiers a naive probabilistic classifier that assumes feature independence given the class (known as Naive Bayes), and a support vector machine (SVM) with a polynomial of degree one as the kernel function. The clustering algorithm follows the Expectation Maximization (EM) technique. The number of clusters is estimated using cross-validation. On each run we use $50\%$ of the examples for training, $25\%$ for validation, and $25\%$ for testing. Reports on accuracy are the average of ten runs (over the testing set). An asterisk at the top right of a number implies the difference is significant at the $p = 0.01$ level (using a $t$-student distribution). Our datasets can be obtained from the University of California at Irvine Repository [1]

### 4.1 Results

Table 1 displays our results. The second column reports the mean accuracy of Naive Bayes, and the third column reports the increase in accuracy using our proposed approach. The fourth and fifth columns show the corresponding performance and increase in accuracy for the linear classifier. There is a clear gain in performance with the probabilistic classifier. The average increase in accuracy for Naive Bayes is $4.56\%$. There is statistically significant enhancement met in eight domains. No performance improvement indicates our algorithm has merged all clusters of each class into a single cluster; such configuration is equivalent to the original dataset and thus produces no change in performance. Compared to previous work [6], we find that merging clusters into single classes (Step 2, Section 3) is indeed effective, but the need for a validation step reduces the number of available training examples, resulting in some performance impact.

The average increase in accuracy for the linear classifier is of $0.89\%$. Performance is almost the same between our proposed approach and the standard version, except for two domains (vehicle and vowel) where the difference is significant. In some cases there is a decrease in performance (not significant), indicating an apparent increase in accuracy on the validation set, but an actual decrease on the testing set.

**Table 1. Predictive accuracy on real-world domains.**

| Domain | Naive Bayes | $\Delta$ Acc | Linear Classifier | $\Delta$ Acc |
|---|---|---|---|---|
| Anneal | 82.21 | 9.98* | 88.13 | 0.42 |
| Balance-Scale | 89.88 | 0.0 | 87.83 | 0.0 |
| Breast-Cancer | 73.52 | 0.38 | 72.96 | 0.63 |
| Breast-W | 96.19 | 0.16 | 96.87 | 0.0 |
| Colic | 79.44 | 1.92 | 83.39 | $-0.22$ |
| Credit-a | 77.87 | 3.71* | 85.04 | 0.15 |
| Credit-g | 78.17 | 0.0 | 76.02 | $-0.23$ |
| Diabetes | 76.81 | 0.0 | 78.07 | 0.0 |
| Heart-c | 83.40 | 0.44 | 85.96 | $-0.53$ |
| Heart-h | 86.84 | 0.0 | 83.48 | 0.61 |
| Hypothyroid | 90.22 | 1.02 | 72.88 | 0.63 |
| Letter | 59.68 | 1.04* | 56.23 | 0.05 |
| Mushroom | 92.91 | 6.45* | 99.93 | 0.04 |
| Segment | 80.18 | 8.57* | 91.88 | 0.0 |
| Sick | 86.36 | 4.82* | 92.19 | 0.0 |
| Soybean | 90.17 | 0.00 | 94.11 | 0.00 |
| Splice | 96.88 | 0.00 | 93.99 | 0.00 |
| Vehicle | 42.52 | 25.24* | 66.86 | 4.88* |
| Vote | 89.17 | 6.02* | 97.49 | $-0.09$ |
| Vowel | 59.92 | 21.53* | 72.17 | 11.53* |

## 5 Summary and Future Work

We propose an approach to improve the accuracy of simple classifiers through a pre-processing step that applies a clustering algorithm over examples belonging to the same class. We demonstrate the resulting knowledge can be exploited to improve the quality of the class decision boundaries. Our algorithm explores the space of possible class assignments over the induced clusters searching to maximize accuracy. Our experiments show that our proposed approach results in either equal or increased accuracy on most of the real-world domains used for analysis.

Future work will try to explain why performance improvement is evident in the probabilistic classifier but not in the linear classifier. We will also look for ways to improve the computational cost of finding the best class-assignment configuration.

## References

[1] C. Blake and M. C.J. UCI, Repository of machine learning databases. *University of California, Irvine, Dept. of Information and Computer Sciences*, 1998.

[2] L. Breiman. Bagging predictors. *Machine Learning Journal*, 24:123–140, 1996.

[3] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. pages 148–156, 1996.

[4] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, pages 1–58, 1992.

[5] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning; data mining, inference, and prediction*. Springer-Verlag, 2001.

[6] R. Vilalta and I. Rish. A decomposition of classes via clustering to explain and improve naive bayes. *14th European Conference on Machine Learning*, 2003.