An Empirical Study of the Suitability of Class Decomposition for Linear Models: When Does It Work Well?

Francisco Ocegueda-Hernandez* Ricardo Vilalta*

Abstract

The presence of sub-classes within a data sample suggests a class decomposition approach to classification, where each subclass is treated as a new class. Class decomposition can be effected using multiple linear classifiers in an attempt to outperform a single global linear classifier; the goal is to gain in model complexity while keeping error variance low. We describe a study aimed at understanding the conditions behind the success or failure of class decomposition when combined with linear classifiers. We identify two relevant data properties as indicators of the suitability of class decomposition: 1) linear separability; and 2) class overlap. We use well-known data complexity measures to evaluate the presence of these properties in a data sample. Our methodology indicates when to avoid performing class decomposition based on such data properties. In addition we conduct a similar analysis at a more granular level for data samples marked as suitable for class decomposition. This extra analysis shows how to improve in efficiency during class decomposition. From an empirical standpoint, we test our technique on several real-world classification problems; results validate our methodology.

Keywords Class Decomposition, Model Selection, Meta-Learning

1 Introduction

There is a broad spectrum of successful applications where the learning algorithm employed for a classification task is limited to linear classifiers, e.g. document classification [2, 20], biomedical work [15], face recognition [3], bioinformatics [16], etc. Linear classifiers have the advantage of keeping the variance component of error low, but may result in high bias (e.g., under non-linear class distributions). One approach that has been extensively studied in recent years is the use of a combination of linear classifiers to replace a single global linear classifier [11, 19, 7, 8, 4, 21, 6, 5, 18]. Such approach increases model complexity, while keeping variance under control.

*Department of Computer Science, University of Houston, Houston Texas, USA.{ocegueda, vilalta}@cs.uh.edu

One instance of compound linear classifiers is that of class decomposition via clustering; here classes are separated into clusters as a pre-processing step to classification. As an illustration, Figure 1 depicts a two-dimensional input space where examples belong to two classes. The dotted line is the decision boundary built by a single global linear classifier. Now, assume a clustering algorithm separates each class into two clusters, whereby we relabel every example to encode class and cluster label; the resulting dataset has now four different classes. The dashed lines are the decision boundaries correctly separating the new four classes. The added flexibility gained by combining linear models does not come with a drastic increase in variance.

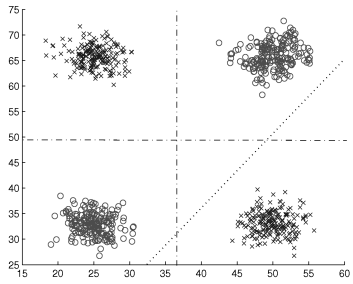
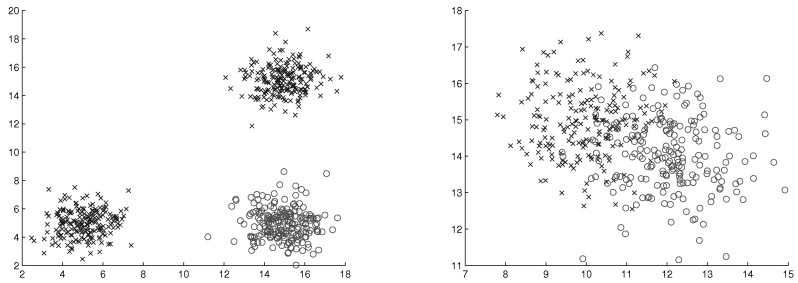


Figure 1: XOR dataset.

Several successful case studies employing class decomposition in conjunction with linear classifiers can be found in the literature [19, 6, 5, 18]. Despite promising results, there is a large number of classification problems where a combination of linear classifiers through class decomposition comes unwarranted. For example, Figure 2 depicts two distributions where a combination of linear classifiers does not bring an advantage over a single linear classifier. Both distributions are two-dimensional and have two classes. The data distribution shown in Figure 2(a) is linearly separable; although one class (class "x") can be divided into two sub-classes, a single linear classifier suffices to separate both classes. Figure 2(b) shows the case where high Bayes error (i.e., high class overlap) obviates any type of class decomposition.



(a) (b)

Figure 2: Data distributions not suitable for class decomposition. (a) Data is linearly separable. (b) Classes overlap significantly, i.e., data exhibits high Bayes error.

Given the potential use of class decomposition in scenarios where linear classifiers have proved effective (e.g., text mining, bioinformatics applications, etc.), but where data abounds, it is important to avoid the extra computational cost incurred by the pre-processing (i.e., clustering) step, especially when the problem is not a good fit for class decomposition. Moreover, among problems that do indeed need the pre-processing step, it would be desirable to automatically select the classes that need such decomposition, instead of blindly applying the step to all classes.

Two important questions are as follows: (1) when is a distribution suitable for class decomposition? (i.e., when does the combination of linear classifiers produced by class decomposition outperform a single global linear classifier?); (2) if a data distribution is suitable for class decomposition, which classes should be decomposed? Our study attempts to answer the questions above by extracting data characteristics from samples by means of data complexity measures; we wish to understand the conditions for success or failure during class decomposition (when used in conjunction with linear classifiers). Specifically, we address the limitations of a combination of linear classifiers obtained by means of class decomposition that reveal two data properties: 1) linear separability between classes; 2) high overlap between classes.

This paper is organized as follows. Section 2 describes related work. Section 3 provides preliminary information on classification, decomposition, and complexity measures. Section 4 reports on our experimental analysis. Section 5 explains our methodology to determine the suitability of class decomposition. Finally, Section 6 gives a summary and discusses future work.

2 Related Work

Several studies have demonstrated that using linear classifiers and class decomposition via clustering together, can improve predictive accuracy when compared to the use of a single global linear classifier. In [18], an empirical study of class decomposition shows a couple of clear successful cases when using linear classifiers. A similar study reported by [6] elaborates deeper on the effect of the number of clusters employed during class decomposition, but results are not conclusive to determine if increasing the number of clusters is beneficial. More recently, [19, 5] successfully applies class decomposition to deal with the class imbalance problem; however, the analysis lacks insight about data properties that favor the decomposition of a single majority class to eliminate class imbalance. A common conclusion among these studies is that class decomposition hardly improves performance when used in conjunction with complex classifiers (e.g. decision trees, support vector machines with radial basis functions or high order polynomials as kernels, etc.). Class decomposition is useful to enhance low variance classifiers exclusively (e.g., naive Bayes, linear classifiers). Previous work fails to elucidate data properties that favor the utilization of class decomposition with linear classifiers. Moreover, there are no guidelines pointing to the classes that seem more favorable for decomposition; in almost all cases, class decomposition is applied indiscriminately over all classes. In this study we address these challenges by exploiting information that lies in the data itself, using a set of well-known data complexity measures.

3 Preliminaries

3.1 Basic Notation in Classification. Let $(A_1, A_2, \cdots, A_n)$ be an n-component vector-valued random variable, where each A_i represents an attribute or feature; the space of all possible attribute vectors is called the 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 $T = \{(\mathbf{x}, y)\}$, $|T| = N$, where $\mathbf{x} = (a_1, a_2, \cdots, a_n)$ is a vector or point in the input space and y is a point in the output space. We assume T consists of independently and identically distributed (i.i.d.)

examples obtained according to a fixed but unknown joint probability distribution in the input-output space $\mathcal{X} \times \mathcal{Y}$. The outcome of the classifier is a function f (or hypothesis) mapping the input space to the output space, $f : \mathcal{X} \to \mathcal{Y}$. We consider the case where a classifier defines a discriminant function for each pairwise coupling of classes $g_{ij}(\mathbf{x})$, $i, j = 1, 2, ..., k$ where $i \neq j$, and chooses the class corresponding to the discriminant function with highest value (ties are broken arbitrarily).

3.2 Class Decomposition via Clustering Class decomposition via clustering is a pre-processing step that has been successfully used to enhance the performance of linear models [18, 6]. It works by partitioning each class into clusters, and by relabelling examples comprised by each cluster with a new class. This decomposition attempts to discover the intrinsic local distribution of subclasses for each class, which can be seen as an indicator of problem complexity. Knowledge about problem complexity can then lead to selecting an appropriate classifier, and has proved to be particularly useful for linear models. In particular, class decomposition allows increasing the capacity of linear classifiers through a piecewise model-building approach. Let's revisit Figure 1, where we show a two dimensional input space with two classes, and where each class has two subclasses; the subclass distribution follows the XOR concept. Clearly, a simple linear classifier is inappropriate here. In contrast, the combination of linear models, where each model separates a subclass from the rest, allows the construction of a more flexible (compound) decision boundary. Class decomposition enable us to cope with distributions where classes spread over disparate regions of the input space.

The mechanism for class decomposition used in this study comprises the following steps: 1) separate the training data T into sets of examples of the same class. That is, T is separated into subsets $T = \{T_j\}$, where each T_j comprises all examples in T labelled with class y_j , $T_j = \{(x, y) \in T \mid y = y_j\}$. 2) for each subset T_j, a clustering algorithm is applied to find clusters of examples grouped together according to some distance metric over the input space. Let c_i^j be the set of such clusters. We map each T_j into a new set T'_j by renaming every class label to indicate not only the class, but also the cluster for each example. 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. In that case, $T'_j = (x, y'_j)$, where $y'_j = (y_j, i)$ whenever example x is assigned to cluster c_j^i. Finally each new subset T'_j is simply the union of all sets of examples of the same class relabelled according to the cluster to which each example belongs, $T' = \bigcup_{j=1}^{k} T'_j$.

3.3 Data Complexity Measures Complexity analysis applied to classification has been extensively studied from a theoretical perspective. Recently, an empirical approach has generated considerable research interest [10, 1, 17, 12, 13]. Much of this research has employed a set of measures to characterize data complexity during classification by means of geometrical and topological properties. The idea is to establish a connection between data characterization and classifier performance. In [10], three data complexity measures are introduced: (1) class overlap measured according to feature discriminatory power; (2) class separability measured according to the length and linearity of the class boundary, and (3) geometry, topology and density of classes, which assume the problem is composed of several manifolds spanned by each class; the shape, position, and interconnectedness of these manifolds provide hints on how well the classes are separated, and on the density or population of each manifold. The complexity of a task is characterized by these measures. Such characterization provides a quantitative perspective to study the learnability of class boundaries, and is of great relevance for operational guidance during model selection. When datasets are characterized by these measures, similar properties are expected to correlate with problems of similar complexity.

In this paper, we employ data complexity measures to analyze the conditions under which class decomposition via clustering improves predictive performance (using linear classifiers). We briefly describe the set of complexity measures employed for our experiments.

Training error of a linear classifier (L2). Linear class separability can be estimated by computing the training error of a linear classifier. Let $g(x)$ be a linear classifier, we define $L2$ as:

$$\mathrm{L2(T)} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{I}_{\{\mathrm{g(x_i)} \neq \mathrm{y_i}\}} \tag{3.1}$$

where $I_{\{\cdot\}}$ is an indicator function. A value of zero in L2 indicates the problem is perfectly linearly separable.

Ratio of average intra/inter class nearest neighbor distance (N2). We compare the within-class example separation with the example separation across classes. For each input instance (x_i, y_i), we compute the following:

$$\mathrm{D_I(x_i, y_i)} = \min_{\mathrm{(x,y)} \in \mathrm{T} | \mathrm{y} = \mathrm{y_i}} dist(x_i, x) \tag{3.2}$$

$$\mathrm{D_O(x_i, y_i)} = \min_{\mathrm{(x,y)} \in \mathrm{T} | \mathrm{y} \neq \mathrm{y_i}} dist(x_i, x) \tag{3.3}$$

$$N2(T) = \frac{\sum_{i=1}^{N} \mathrm{D_I}(\mathrm{x_i}, \mathrm{y_i})}{\sum_{i=1}^{N} \mathrm{D_O}(\mathrm{x_i}, \mathrm{y_i})} \tag{3.4}$$

where $dist()$ is a distance function (Section 4.1). Low values for $N2$ suggest examples of the same class lie closely in the feature space. High values indicate high dispersion among examples of the same class.

Leave-one-out error rate of the one-nearest neighbor classifier (N3). We capture the relative closeness of examples from different classes, by computing the leave-one-out error rate of the one-nearest neighbor classifier (the kNN classifier with $k = 1$, or 1NN). Low values point to a large margin.

4 Empirical Study

We begin our study by first describing an empirical analysis of the use of class decomposition in conjunction with linear classifiers using several real-world datasets. We postpone analysis of the suitability of class decomposition to Section 5. We describe a number of experiments to compare the classification performance of several classifiers, including a single global linear classifier, a classifier with high capacity, and several composite classifiers obtained through different settings during class decomposition. We also compute data complexity measures (i.e., $L2$, $N2$ and $N3$) to correlate data properties with classification performance.

4.1 Experimental Setup In all our experiments, we employ k-means as the clustering algorithm for class decomposition. We limit the value of k to the set $\{2, 3, 4, 6, 10\}$. Implementations of our learning algorithms come from WEKA machine-learning class library [9] using default values. Our data complexity measures come from the data complexity library DCoL [1] [14]. We use the Heterogeneous Value Difference Metric (HVDM) as the distance metric (with options -cM 3 and -nM 2). In order to facilitate the reproducibility of our experiments, we use our own modified version of WEKA[2].

4.2 Experimental Datasets We use several datasets from the UCI Machine Learning Database Repository (http://www.ics.uci.edu/mlearn/). Table 1 displays the datasets employed in this study. The datasets stand as representative examples of success and failure in the use of class decomposition for linear classifiers based on results reported in [18, 6, 19].

Table 1: Datasets and properties

Id.	Dataset	Examples	Attributes	Classes
1	Vowel	990	13	11
2	Vehicle	846	18	4
3	PenDigit	10992	16	10
4	SatImage	6435	36	6
5	Image	2310	19	7
6	Heart-c	303	13	2
7	Credit-g	1000	20	2

4.3 Evaluation of Classification Performance We report on a series of experiments to identify datasets suitable for class decomposition. We evaluate predictive accuracy for each dataset using the following classifiers: i) a single global linear classifier using Support Vector Machines with a linear kernel (SMO); ii) five composite linear classifiers generated by the use of class decomposition with $k = 2, 3, 4, 6, 10$ (CD2, CD3, CD4, CD6 and CD10 respectively); and iii) a high-capacity classifier using kNN with $k = 1$ (1NN). SVMs are trained using a one-vs-one approach having the same misclassification cost for each class. Multi-class datasets are separated into a set of pairwise coupling classification datasets. On each dataset, we report the average of a 10-fold cross validation.

Table 2 displays our results. The first column describes the dataset used for our experiments. The second column reports on the accuracy of the linear classifier (SMO). The third to seventh columns show accuracy for the composite classifiers (CD2, CD3, CD4, CD6 and CD10). The eight column shows the accuracy of the high-capacity classifier (1NN). Numbers enclosed in parentheses represent standard deviations. A significant difference with respect to the linear classifier (SMO) is shown in bold. Our tests of significance assume a t-student distribution with a 95% confidence level.

Our results show only two datasets suitable for class decomposition, namely, PenDigit and SatImage; in these cases, class decomposition shows a significant gain in performance with respect to the linear classifier. For the PenDigit dataset, class decomposition using $k = 2$ is sufficient to achieve a significant gain. For the SatImage dataset, a significant improvement is obtained using $k = 6$. No performance improvement is observed in all other datasets. The high-capacity classifier (1NN) significantly outperforms the linear classifier (SMO) on 4 datasets, namely, Vowel, PenDigit, SatImage and Image. We observe that in the two

[1] DCOL is available at http://dcol.sourceforge.net

[2] We have made our modified WEKA tool available at http://www2.cs.uh.edu/~ocegueda/tools/clsdcomp.htm

Table 2: Accuracy results on experimental datasets using a single global linear classifier (SMO), five composite linear classifiers generated by the use of class decomposition (CD2, CD3, CD4, CD6 and CD10), and a Nearest Neighbor (1NN) classifier.

Id.	SMO	CD2	CD3	CD4	CD6	CD10	1NN
1	71.41 (2.98)	73.03 (6.53)	66.26 (4.29)	61.62 (2.86)	54.65 (3.67)	42.83 (7.36)	**99.29 (0.83)**
2	74.36 (5.48)	72.47 (4.14)	72.71 (3.50)	71.76 (3.18)	67.74 (6.27)	61.95 (5.30)	69.86 (4.47)
3	97.96 (0.36)	**99.10 (0.23)**	**99.18 (0.24)**	**99.22 (0.32)**	**99.20 (0.31)**	**99.24 (0.32)**	**99.36 (0.17)**
4	86.85 (0.62)	86.31 (0.92)	87.12 (0.74)	88.00 (1.00)	**88.86 (0.98)**	**89.48 (0.93)**	**90.21 (1.16)**
5	93.07 (1.66)	91.73 (1.78)	93.25 (0.98)	93.46 (1.05)	93.20 (1.19)	92.47 (1.29)	**97.14 (0.62)**
6	83.80 (6.80)	80.18 (5.84)	82.16 (5.30)	81.47 (8.24)	79.85 (8.95)	77.55 (7.12)	75.88 (5.06)
7	75.10 (3.45)	73.00 (3.68)	70.90 (3.48)	71.30 (3.95)	71.10 (5.04)	72.30 (4.60)	72.00 (3.09)

datasets where 1NN outperforms SMO but CD does not (i.e., Vowel and Image), there is an inflection point (k^*) in the k value (i.e., maximum performance). k values greater than k^* showed a decrease in performance. For the Vowel dataset $k^* = 2$ and for the Image dataset $k^* = 4$. This result may seem counter-intuitive, because the greater the k, the more similar the behavior of the composite classifier to 1NN. That is, if 1NN is able to outperform the linear classifier, then we would expect that increasing k during class decomposition would produce a corresponding increase in performance. Finally, we observe that in two datasets (Heart-c and Credit-g), neither class decomposition nor 1NN outperforms the linear classifier. The loss in classification performance between SMO and 1NN serves as an indicator of data properties that lead to failure in the use of class decomposition.

4.4 Evaluation of Data Complexity Measures. Our experiments aim at identifying the presence of two data properties: linear separability and class overlap. Complexity measures are obtained through two steps: i) transformation of a multi-class classification problem into a set of 2-class classification problems (pairwise coupling). If there are k-classes, we build $\frac{k\times(k-1)}{2}$ two-class datasets. ii) computation of data complexity measures for each two-class problem generated in step (i); we compute $L2$, $N3$, and $N2$ described in Section 3.3, and an estimation of the difference in classification performance between a linear classifier (SMO) and a higher complexity classifier (NN), computed as $L2 - N3$.

Table 3 displays our estimations. The first column describes the dataset used for our experiments. The next columns (2-5) report on average values for $L2$, $N3$, $L2 - N3$, and $N2$. The sixth column captures the presence of linear separability; if $L2 = 0$ then Yes, otherwise No. The seventh column indicates low or high presence of class overlap; if $N2 <= 0.5$ then Low, otherwise High. The eighth column (CD) is marked as (+) if there is presence of both non-linear separability and low overlap, otherwise it is marked as (−). The (+) mark suggests the use of class decomposition.

5 On the Suitability of Class Decomposition

We now analyze the suitability of class decomposition under linear classifiers, using the results from our empirical study (see Section 4).

5.1 Dataset Properties. We focus first on dataset properties, and introduce a set of relevant definitions for our analysis. Let T be a data sample ($T \subset \mathcal{X} \times \mathcal{Y}$), let $M(T)$ be the optimal (along parameter k) compound model that arises from using class decomposition in conjunction with linear classifiers in T, and let $L(T)$ be a single linear classifier obtained from T.

DEFINITION 5.1. *T is class decomposable, if $R(M(T)) < R(L(T))$ where $R(\cdot)$ is the risk or generalization error of a classifier.*

DEFINITION 5.2. *T is linearly separable, if $L2(T) = 0$.*

We start our analysis through a concrete example. Let us denote the sample shown in Figure 2(a) as T_l; it is clearly linearly separable[3] (i.e., $L2(L(T_l)) = 0$), but it is not class decomposable (i.e., $R(M(T_l)) \geq R(L(T_l))$). In general, we should avoid the use of class decomposition for linearly separable classification problems because there is no guarantee of classification improvement.

We now turn to another concrete example where the data sample is not linearly separable. Let us represent as $T_{\bar{l}}$ the sample shown in Figure 1. Here $L2(T_{\bar{l}}) > 0$), but the sample is class decomposable (i.e., $R(M(T_{\bar{l}})) < R(L(T_{\bar{l}}))$). We observe in Figure 1, that the data is separable, meaning Bayes error is negligible. This fact allows us to gain insight about the conditions for success in the use of class decomposition. Specifically, when

[3]We note that our definition for linear separability is limited to T; the training error is frequently an optimistic approximation to the true error.

Table 3: Averages of data complexity measures (per dataset). LS stands for linearly separability, and CD stands for class decomposition suitability (the + mark suggests the dataset is suitable for the use of class decomposition).

Id.	L2	N3	L2-N3	N2	LS	Overlap	CD
1	0.1021	0.0000	0.1021	0.2070	No	Low	+
2	0.2423	0.1208	0.1215	0.5760	No	High	−
3	0.0086	0.0011	0.0075	0.2141	No	Low	+
4	0.0410	0.0217	0.0193	0.3839	No	Low	+
5	0.0229	0.0067	0.0162	0.1773	No	Low	+
6	0.1820	0.2570	-0.0750	0.7570	No	High	−
7	0.2970	0.3390	-0.0420	0.8690	No	High	−

a sample is not linearly separable, but it is separable through increased model capacity, then the use of class decomposition is beneficial.

Following the analysis above, we now look into class separability without the restriction for linearity. We make use of the Nearest Neighbor(NN) algorithm, particularly, data complexity measure $N3$. We introduce the following definition:

DEFINITION 5.3. *T is said to be separable, if $N3(T) = 0$.*

We can now state that $T_{\bar{l}}$ is not linearly separable (i.e., $L2(T_{\bar{l}}) > 0$) but it is separable (i.e., $N3(T_{\bar{l}}) = 0$); and as expected $T_{\bar{l}}$ is class decomposable (i.e., $R(M(T_{\bar{l}})) < R(L(T_{\bar{l}}))$).

The example above enables us to move forward to the general case where $L2(T) > 0$ and $N3(T) > 0$. We stated that when a sample is not linearly separable, but Bayes error is negligible, an opportunity exists for improvement using class decomposition. We argue such scenario can be captured by looking at a positive difference in the classification performance between a linear classifier and a high-capacity classifier. We make the following definition:

DEFINITION 5.4. *The complexity error gap when using class decomposition in T is $E(T) := L2(T)- N3(T)$.*

In our definition, we take the performance of a nearest neighbor algorithm (1NN) as an approximation to the classifier with best case performance using class decomposition. The rationale behind this is to establish an optimistic upper bound in performance improvement.

We now introduce the definition of *reducible error*, to understand when to expect a performance gain using class decomposition.

DEFINITION 5.5. *T is said to have a reducible error, if $E(T) > 0$.*

The definition enables us to understand the advantage that comes when generating a combination of linear classifiers through class decomposition. Specifically, linear classifiers exhibit high bias and low variance, and as such are less affected by noisy data, i.e., they are stable. If the error difference between a linear classifier and 1NN is negative, then the likelihood of noisy data increases, producing an irrecoverable error for 1NN. Class decomposition is not appropriate here. As an illustration, Figure 2(b) shows a two-dimensional sample characterized by a high degree of overlap between the two classes. Class decomposition does not result in any improvement because of high Bayes error. To avoid the use of class decomposition in this type of scenario, we will say that T has low class overlap, if $N2(T) < \gamma$, where $0 < \gamma < 1$ will be a user-defined constant.

We are now ready to define data properties favorable for class decomposition. Previous work [18, 6, 19] suggests the presence of a heterogeneous class distribution (i.e., a distribution where sub-classes are widely spread over the feature space) as a condition for the success in the use of class decomposition. Previous work fails to detect the presence of this condition in a sample. We propose a simple and measurable condition to identify the success of class decomposition by verifying the presence of two data properties: i) there is a positive difference between the error of a linear classifier and the error of a high-capacity classifier (i.e., there is room for improvement when we increase model complexity by means of class decomposition); and ii) class overlap or Bayes error is minimal. We formalize these conditions with the following proposition:

PROPOSITION 5.1. *A data sample T is class decomposable if T meets the following two conditions:*

(i) T has reducible error.

(ii) T has low class overlap.

We now return to our empirical study to validate our proposition. The eighth column (CD) in Table 3 marks as (+) those samples that are class decomposable, and as (-) otherwise. Vowel, PenDigit, SatImage and Image datasets are marked as (+); these datasets are

Table 4: Complexity measures (L2, N3, L2-N3, and N2) and accuracy per class for a single global linear classifier (SMO), and the best linear composite classifier using class decomposition (CD*) for the Vowel dataset. Diff. stands for the accuracy difference ($CD^* - SMO$) and #Cls. stands for the number of clusters employed by CD*.

Id (Class Name)	L2	N3	L2-N3	N2	SMO	CD*	Diff.	#Cls.
1 (hid)	0.0979	0.0000	0.0979	0.1548	98.50	99.56	1.06	2
2 (hId)	0.1274	0.0000	0.1274	0.1968	95.22	98.61	3.39	3
3 (hEd)	0.1002	0.0000	0.1002	0.2000	97.00	99.33	2.33	3
4 (hAd)	0.0837	0.0006	0.0831	0.1914	98.28	99.39	1.11	2
5 (hYd)	0.0939	0.0006	0.0933	0.2139	96.50	96.78	0.28	2
6 (had)	0.1488	0.0018	0.1470	0.2427	93.22	96.39	3.17	2
7 (hOd)	0.1331	0.0006	0.1325	0.2177	96.39	98.72	2.33	2
8 (hod)	0.0612	0.0006	0.0606	0.1946	97.95	98.56	0.61	3
9 (hUd)	0.0945	0.0000	0.0945	0.2573	95.83	97.39	1.56	2
10 (hud)	0.0688	0.0000	0.0688	0.1908	98.44	99.00	0.56	3
11 (hed)	0.1145	0.0006	0.1139	0.2168	96.78	99.00	2.22	2

characterized by having both a reducible error and low class overlap (γ was set to 0.5). Under these two data conditions, class decomposition showed to be beneficial when compared to a single linear classifier, as shown in Table 2. We note that only in two datasets (PenDigit and SatImage) is this difference significant. Samples marked as (-) are rejected for various reasons. Vehicle dataset is rejected because $N2 > 0.5$, regardless of the difference in $L2 - N3$. For Heart-c and Credit-g datasets, the difference $L2 - N3$ is indeed negative.

Results show the value of Proposition 5.1 for the successful use of class decomposition. Specifically, when omitting significant differences, sensitivity and specificity are 100% for (+) and (-) classes; under significant differences, sensitivity is 100% for class (+) but there is a decrease in the specificity for class (-) to 60%. From a practical standpoint, Proposition 5.1 can be employed to identify the cases where no decomposition is recommended. For cases when it is recommended, an additional analysis can be effected to improve in computational efficiency as described next.

5.2 A Deeper Analysis: Improving Efficiency. In this section we deepen our analysis by looking at learning performance on individual classes. Our methodology follows the next steps: i) for each class C_i, we define a new set of subproblems, one for every pairwise coupling of C_i and all other classes; subproblems are stored under the set S_{C_i}; ii) compute averages for $L2$, $N3$, $L2 - N3$ and $N2$ in S_{C_i}; iii) for each sub-problem in S_{C_i}, create the corresponding k-decomposition via clustering using the following values for $k = 2, 3, 4, 6, 10$, and evaluate predictive accuracy for a single global linear classifier and a composite linear classifier generated through class decomposition (with k clusters per class). For each sub-problem, report on accuracy estimated through 10-fold cross validation.

Table 4 displays results for the Vowel dataset (a dataset reported as a successful case for decomposition in two previous reports [18, 6]). The first column describes the class under analysis. The 2nd-5th columns report on the average value of $L2$, $N3$, $L2-N3$, and $N2$ respectively. The sixth column reports on the accuracy of the linear classifier (SMO). The seventh column reports on the accuracy for the best composite classifier obtained among the different values of k using class decomposition (CD*). The eighth column reports the difference in accuracy between (SMO) and (CD*). The ninth column shows the number of clusters employed by the best linear composite classifier.

If for each class in Table 4 we apply Proposition 5.1 to predict the suitability of class decomposition, we would observe all classes labeled as (+) (i.e., all subproblems are suitable for class decomposition). The eighth column shows a positive value on every class. This type of analysis can be employed to select just a subset of classes for decomposition. We can, for example, sort all sub-problems in descending order based on the value of the eighth column (i.e., the difference in performance between SMO and CD*), and iteratively add classes until we reach an inflection point. Specifically, we begin by decomposing the class with highest performance difference (8th column, Table 4), and keep adding more classes (in decreasing order, 8th column) until all classes are included.

Figure 3 shows accuracy when incorporating this iterative approach. As a reference, we also show performance for both the single linear classifier (SMO) and the best class decomposition classifier (CD*). As can be seen in Table 4, class 2 corresponds to the highest performance difference. In principle, we could decompose this class only and still achieve good results (Figure 3).

In summary, a simple analysis can show a reduced number of classes in true need for decomposition.

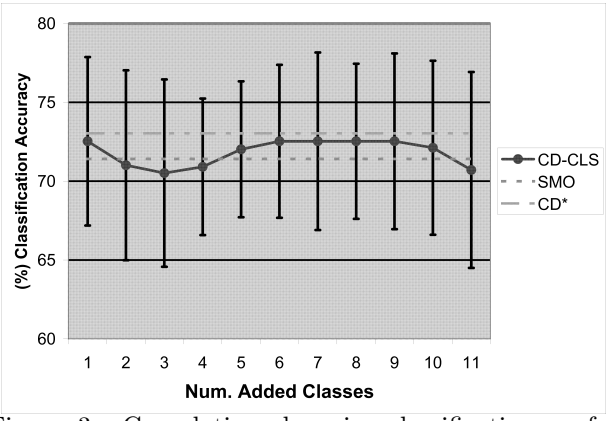


Figure 3: Cumulative classwise classification performance for Vowel Dataset

In order to verify that these results are consistent with our previous analysis, we report on the same type experiment for the PenDigit Dataset. Results are given in Table 5 and Figure 4. We observe in Figure 4, that the maximum performance value is reached using only 6 classes, namely, $1, 2, 6, 8, 9, 10$. This analysis saves on the computational cost of decomposing four additional classes.

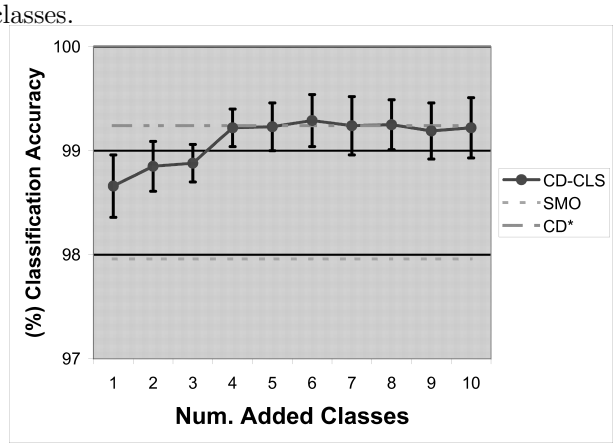


Figure 4: Cumulative classwise classification performance for PenDigit Dataset

6 Conclusions and Future Work

We describe a study aimed at understanding under what conditions a data sample is suitable for class decomposition (using linear classifiers). We focus our methodology in the identification of two data properties: 1) linear separability using $L2$ and $N3$ measures; and 2) class overlap using $N2$ measure. Our analysis is based on experiments performed on seven real-world domains; results support the following conclusions: a) if the classification problem is linearly separable ($L2 = 0$), then class decomposition should be avoided; b) if the classification problem is not linearly separable ($L2 > 0$), and a high-capacity model is able to separate the classes ($N3 = 0$), then this sample is suitable for class decomposition; c) if the classification problem is not linearly separable ($L2 > 0$) and a high-capacity model is not able to separate the classes ($N3 > 0$), then we propose using the difference $L2 - N3$ as an indicator for the use of class decomposition. If $L2 - N3 <= 0$ then class decomposition should be avoided because there is no room for performance improvement. If $L2 - N3 > 0$ the use of class decomposition can be effected when such a difference is above an user-defined threshold value (defined by γ). d) If there is high overlap between classes (i.e., $N2$ is above a threshold), then class decomposition should be avoided because there is no guarantee for improvement. In addition, the proposed method can be applied at a more granular level to identify those classes specifically suitable for decomposition. This last method adds in computational efficiency by avoiding class decomposition when unnecessary.

Our research exhibits the following limitations: 1) the use of a nearest neighbor algorithm carries high computational cost; and 2) threshold values are chosen empirically. As future work we plan on tackling these limitations as follows: 1) we plan to use an approximation to $N3$ with a lower computational cost, by computing the fraction of examples lying along the class boundary by means of a minimum spanning tree; 2) we plan to perform a similar analysis of $N2$ as in [17] to set the $N2$ threshold according to the dimensionality and the type of attributes (e.g., nominal, numeric or mixed).

References

[1] Mitra Basu and Tin Kam Ho, editors. *Data Complexity in Pattern Recognition*. Springer-Verlag, London, 2006.

[2] Kai-Wei Chang and Dan Roth. Selective block minimization for faster convergence of limited memory large-scale linear models. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA*, pages 749–754, August 2011.

[3] Nitesh V. Chawla and Kevin W. Bowyer. Designing multiple classifier systems for face recognition. In *Proceedings of the 6th International Conference on Multiple Classifier Systems, Seaside, CA, USA*, pages 407–416, June 2005.

[4] Feng Chen, Chang-Tien Lu, and Arnold P. Boedihardjo. On locally linear classification by pairwise coupling. In *Proceedings of the 8th IEEE International*

Table 5: Complexity measures (L2, N3, L2-N3, and N2) and accuracy per class for a single global linear classifier (SMO), and the best linear composite classifier using class decomposition (CD*) for the PenDigit dataset. Diff. stands for the accuracy difference ($CD^* - SMO$) and #Cls. stands for the number of clusters employed by CD*.

Id (Class Name)	L2	N3	L2-N3	N2	SMO	CD*	Diff.	#Cls.
1 (0)	0.0086	0.0008	0.0078	0.1681	99.48	99.93	0.45	6
2 (1)	0.0210	0.0015	0.0195	0.2232	99.49	99.78	0.28	6
3 (2)	0.0066	0.0017	0.0049	0.2249	99.82	99.89	0.07	6
4 (3)	0.0045	0.0020	0.0025	0.2368	99.72	99.78	0.06	10
5 (4)	0.0031	0.0005	0.0026	0.2144	99.89	99.95	0.06	3
6 (5)	0.0146	0.0008	0.0138	0.2090	99.33	99.88	0.55	10
7 (6)	0.0010	0.0006	0.0005	0.2047	99.95	99.96	0.02	6
8 (7)	0.0071	0.0013	0.0058	0.2226	99.74	99.89	0.15	3
9 (8)	0.0105	0.0005	0.0100	0.2161	99.36	99.95	0.60	6
10 (9)	0.0091	0.0011	0.0079	0.2209	99.68	99.82	0.14	10

Conference on Data Mining (ICDM), Pisa, Italy, pages 749–754, December 2008.

[5] M. Murat Dundar, Matthias Wolf, Sarang Lakare, Marcos Salganicoff, Raykar, and Vikas C. Polyhedral classifier for target detection: a case study: colorectal cancer. In *Proceedings of the 25th International Conference on Machine learning (ICML), New York, NY, USA,* pages 288–295, 2008.

[6] Dmitriy Fradkin. Clustering inside classes improves performance of linear classifiers. In *Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Dayton, Ohio, USA*, volume 2, pages 439–442, November 2008.

[7] Zhouyu Fu, Antonio Robles-Kelly, and Jun Zhou. Mixing linear svms for nonlinear classification. *IEEE Transactions On Neural Networks*, 21:1963–1975, December 2010.

[8] Kun Gai and Changshui Zhang. Learning discriminative piecewise linear models with boundary points. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI), Atlanta, Georgia, USA*, pages 444–450, July 2010.

[9] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[10] Tin Kam Ho and Mitra Basu. Complexity measures of supervised classification problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):289–300, 2002.

[11] Yujian Li, Bo Liu, Xinwu Yang, Yaozong Fu, and Houjun Li. Multiconlitron: A general piecewise linear classifier. *IEEE Transactions on Neural Networks*, 22(2):276–289, 2011.

[12] Julian Luengo and Francisco Herrera. Domains of competence of fuzzy rule based classification systems with data complexity measures: A case of study using a fuzzy hybrid genetic based machine learning method. *Fuzzy Sets and Systems*, 161(1):3–19, 2010.

[13] Yusuke Nojima, Shinya Nishikawa, and Hisao Ishibuchi. A meta-fuzzy classifier for specifying appropriate fuzzy partitions by genetic fuzzy rule selection with data complexity measures. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 264 –271, June 2011.

[14] Albert Orriols-Puig, Nuria Macia, and T. K. Ho. Documentation for the data complexity library in c++. GRSI 2010001, La Salle Universitat Ramon Llull, Barcelona (Spain), December 2010.

[15] Erinija Pranckeviciene, Richard Baumgartner, and Ray L. Somorjai. Using domain knowledge in the random subspace method: Application to the classification of biomedical spectra. In *Proceedings of the 6th International Conference on Multiple Classifier Systems, Seaside, CA, USA*, pages 962–971, June 2005.

[16] Franck Rapaport, Emmanuel Barillot, and Jean P. Vert. Classification of arraycgh data using fused svm. *Bioinformatics*, 24(13):i375–i382, July 2008.

[17] J. S. Sanchez, R. A. Mollineda, and J. M. Sotoca. An analysis of how training data complexity affects the nearest neighbor classifiers. *Pattern Anal. Appl.*, 10:189–201, July 2007.

[18] Ricardo Vilalta, Murali-Krishna Achari, and Christoph F. Eick. Class decomposition via clustering: A new framework for low-variance classifiers. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM), Melbourne, Florida, USA*, pages 673–676, December 2003.

[19] Junjie Wu, Hui Xiong, and Jian Chen. Cog: local decomposition for rare class analysis. *Data Mining and Knowledge Discovery*, 20(2):191–220, 2010.

[20] Hsiang-Fu Yu, Cho-Jui Hsieh, Kai-Wei Chang, and Chih-Jen Lin. Large linear classification when data cannot fit in memory. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI), Barcelona, Catalonia, Spain*, pages 2777–2782, July 2011.

[21] Jie Zhou, Hanchuan Peng, and Ching Y. Suen. Data-driven decomposition for multi-class classification. *Pattern Recognition*, 41(1):67–76, 2008.