

Clustering Inside Classes Improves Performance of Linear Classifiers

Abstract

This work systematically examines a Clustering Inside Classes (CIC) approach to classification. In CIC, each class is partitioned into subclasses based on cluster analysis. We find that CIC, by extracting local structure and producing compact subclasses, can improve performance of linear classifiers. It is compared against a global classifier on four benchmark datasets, using some of the most popular classification methods such as regularized logistic regression and linear Support Vector Machines. We discuss and empirically analyze the effect of the training set size and the number of clusters per class on the results of the CIC approach. We also examine use of an automated method for selecting the number of clusters for each class.

1 Introduction

The idea of using unsupervised clustering as a supplement to supervised classification has been used informally by the pattern recognition community since early 1960's [25]. In the early 90s, the machine learning community started actively investigating local learning, classifier ensembles, mixtures of experts or input partitioning [10, 26, 18, 21]. All of these approaches involved (implicit or explicit) clustering of points across the classes, i.e. without considering the labels.

A natural question in this context is whether the class label information can be used in the clustering in a way that would improve the classification. In this work we analyze Clustering Inside Classes (CIC) approach, described in [11, 7], that does precisely that. We show that this approach can lead to improved accuracy over the state of the art linear classifiers, such as regularized logistic regression [22, 15] and Support Vector Machines (SVM) [23]. We evaluate the role of dataset characteristics (such as the size of the training set) in obtaining such improvements, as well as the effect of parameters such as number of clusters per class. These results are supported by extensive experimentation on the synthetic data and several benchmark datasets from the UCI repository [1].

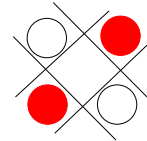


Figure 1. The two classes, represented on a plane with filled and empty circles, each consist of two clusters/subclasses. The lines demonstrate that each cluster/subclass can be easily separated from the others, while the classes themselves are not linearly separable.

This paper has the following structure. In Section 2 we illustrate the ideas behind CIC with an example and describe the algorithm. In Section 3 we describe experimental work on the synthetic and benchmark datasets. In Section 4 we discuss related work, some of the observations and directions for future work. In Section 5 we summarize the main contributions of our work.

2 Clustering Inside the Classes

An intuitive motivation for Clustering Inside Classes approach can be seen in Figure 1. This dataset is not linearly separable - we can't separate these two classes using a single linear classifier in the plane. Clearly, each class consists of two "subclasses." If we knew which training point belongs to which subclass, we would be able to construct accurate simple classifiers for each subclass (indicated by the lines in Figure 1), and correctly classify new points with respect to the original two classes. The apparent problem is that the class labels do not contain information about the subclasses. We can attempt to extract such information by applying methods of cluster analysis to find groups of similar points in each class (clusters, or "subclasses"). Once such clusters are found we can treat them as distinct classes for the purpose of constructing classifiers. Thus, surprisingly, by increasing the number of classes we can get more accurate classification performance.

This example (others like it can be easily constructed) suggests that CIC may be particularly useful are when a class consists of loosely disconnected components or has an "odd" shape. Our experiments on synthetic data support this conclusion. Even when we cannot be certain that these conditions hold (for example due to high dimensionality of the real world data), the CIC approach can improve performance. It is able to do so by partitioning classes into convex subclasses that can be easily separated by linear classifiers. This has the additional benefit of identifying class structure of the data.

2.1 CIC Approach

The CIC classification scheme is described in Algorithm 1. During the training stage each class is partitioned into k clusters (lines 1-3), and then classifier R is trained to classify a new point into one of the resulting clusters. At classification time, a new point is assigned to some cluster i (line 1 or classification stage) which corresponds to a single class label that is returned as prediction (line 2 of the classification stage). When $k = 1$, the algorithm produces a regular K -class classifier.

Note that the scheme described above partitions each class into the same number of clusters, k . Our experimental results will show that even under such a restriction the CIC approach can improve performance of state of the art linear classifiers. However, this restriction is clearly not necessary, since k could be set separately for each class, by the user or automatically. A number of heuristics exist for automatically choosing the number of clusters [16, 5, 19] when applying a clustering algorithm, however there is no single standard approach. We will experiment with automatically selecting the number of clusters in each class using the method of [17] and show that it performs better than the global classifier.

Algorithm 1 Training and Classification with CIC

Require: A set W with $K \geq 2$ classes, an integer $k \geq 1$.
 {Training with CIC}
 1: **for** $j = 1, \dots, K$ **do**
 2: Partition class L_j into k clusters.
 3: **end for**
 4: Train classifier R using all training data to recognize all $k \cdot K$ clusters.
Require: A point x . {Classification with CIC}
 1: Let $i = R(x)$, $i = 1, \dots, k, \dots, k \cdot K$.
 2: Return class of cluster i .

3 Experimental Validation

3.1 Methods

We decided to use K-Means [6, 14] in CIC. This popular hard partitioning method also has an important characteristic of producing convex clusters that should be easily separable. Its run-time is linear in the number of points and clusters, and is usually only a fraction of time required to train a classifier such as SVM on the same data.

Selection of the number of clusters, k , a parameter that has to be specified for K-Means and many other clustering algorithms, is a well-known problem. We experiment with a set of different pre-specified values for this parameter and also with selecting k automatically for each class using the X-Means software [17]. X-Means implements a fast version of K-Means using pre-computed kd-trees and allows automated selection of the number of clusters based on BIC (only the upper limit on k needs to be specified).

We experiment with two linear classifiers, Support Vector Machines and Bayesian Multinomial Regression (BMR). The particular implementation of SVM that we used was LIBSVM v2.71 [3], with a linear kernel (parameter settings "-s 0 -t 0"). The value of the hyperparameter C to be used for each dataset was selected by 5-fold stratified cross-validation on the training set. The values considered were of the form 2^p , with $p = \{-2, -1, 0, 1, \dots, 12\}$. (Such a set of values was previously used in [9]).

BMR¹ implements a multi-class penalized logistic regression classifier [15]. We used version 1.60 of BMR, with a Gaussian prior on the weights, which leads to ridge logistic regression; and BMR's internal cross-validation procedure (10-fold) for selecting the hyperparameter (prior variance, σ^2) from a list of possible values based on maximizing log-likelihood of the training data. The list of potential values was: 0.1, 1, 2.25, 4, 6.25, 9, 12.25, 16, 20.25, 25, 30.25, 36, 42.25, 49, 56.25, 64, 100.

Additionally, we investigated the use of Radial Basis Function (Gaussian kernel) SVM which constructs a non-linear decision surface and is considered to be one of the most powerful classification approaches. Because of its non-linearity, RBF SVM is much less likely to suffer from odd-shaped classes and thus is not expected to benefit from CIC-type approaches. We examine this hypothesis empirically. The value of the hyperparameters C and γ is selected by 5-fold cross-validation on the training set. The values considered for both parameters were of the form 2^p , with $p = \{-2, -1, 0, 1, \dots, 12\}$ for C , and $p = \{-8, \dots, 8\}$ for γ .

¹www.bayesianregression.org/bmr.html

Dataset	C (SVM)	C, γ (RBF SVM)	σ^2 (BMR)
Image	2^4	$2^{10}, 2^{-5}$	42.25
Pendigit	2	$2^3, 2^{-3}$	6.25
Satimage	2^{-1}	$2^7, 2^{-2}$	2.25
Vowel	2^3	$2^8, 2^{-4}$	4

Table 1. Hyperparameter values selected (details in Section 3.1).

3.2 Clustering Inside Classes on Synthetic Data

We have conducted experiments on synthetic data in the past [7]. They suggest that CIC with linear classifiers performs significantly better in the presence of non-convex and non-linearly-separable classes than linear methods alone. They also indicate that having somewhat greater number of clusters than the number of intrinsic components does not hurt the performance (provided that there is sufficient amount of data). Having fewer clusters than components however does not lead to improvements.

We will see that even when we cannot be certain that these conditions hold (for example due to high dimensionality of the real world data), the CIC approach can improve performance. It is able to do so by partitioning classes into convex subclasses that can be easily separated by linear classifiers. This has the additional benefit of identifying class structure of the data.

3.3 Benchmark Datasets

For the experiments we used four well-known datasets from UCI [1]: **Image Segmentation**, **Pendigit**, **Satellite Image (Satimage)** and **Vowel**. They were chosen because they are all multi-class classification problems with real-valued variables, and thus represent a reasonable set for an evaluation of CIC. In all of these datasets, except for Satimage, the relative class sizes are approximately the same in the tests sets as in the training sets.

The SVM and BMR hyperparameters selected for these datasets are specified in Table 1. The properties of these datasets are summarized in Table 2.

All datasets were preprocessed with the conventional statistical normalization: each feature was independently transformed to have zero mean and unity variance.

We decided to use training/test splits described in the UCI repository [1]. This would make our results comparable to most described in the past and would allow future comparisons. However, researchers do not always use the above splits. For example, a widely cited paper [9] on multi-class SVM approaches uses the same split on Satimage, but does 10-fold cross-validation on the full Vowel dataset. Not surprisingly, their results on Vowel are much better than can

be achieved on the partition used here. (Their results on Satimage with RBF SVM are comparable to ours).

3.4 Clustering Inside Classes on Benchmark Data

Tables 3 and 4 shows the accuracy of CIC on each dataset for different number of clusters per class, $k = 1, 2, 3, 4, 5, 6, 8, 10, 15$ and with X-Means. Superscripts ¹, ², ³ will be used to mark results that are significantly different (at 0.95, 0.99 and 0.999 confidence levels respectively) from the performance of the basic classifier ($k = 1$), as determined using McNemar Test, advocated by Dietterich [4] as having low probability of Type I error (i.e. rejecting the null hypothesis when it is correct).

It is clear from these results that on all datasets except for Image CIC leads to improvement in classification accuracy for at least some values of k . With the SVM, the results on the Image dataset with $k > 2$ are significantly worse at 0.99 level than those of the basic classifier. On the other hand, the accuracy of CIC is significantly better at 0.999 level for all values of k (except $k = 15$) on the Pendigit dataset and for $k > 3$ on the Satimage dataset (again, except $k = 15$); and is somewhat better (0.95 level) for $k = 3, 4$ on the Vowel dataset. The results with BMR are qualitatively similar (significant improvements on the Pendigit and Satimage datasets, no significant difference on the Vowel dataset, and worse results on the Image dataset). For $k = 15$ the results are significantly worse than with $k = 1$ on all four datasets with all methods.

Notice that the value of k for which the best results are obtained appears related to the number of points per class in the training set. On the Image dataset this number is on average 30, and the best results are for $k = 1$ (basic classifier). For the other, larger, datasets the performance improves as k increases up to a point, and then starts to decrease. Intuitively, increasing the number of clusters leads to clusters with fewer point, which in turn makes it difficult to train good classifiers for distinguishing them. This argument is both intuitively accessible and can be supported by learning theory [23]. However, as we will show later, these differences in performance are not completely due to the training set size, but are also affected by the intrinsic structure of the classes. In other words, Image dataset has classes that consist of a single component, while the other datasets have classes with a more complicated structure.

Using X-Means does not give the best results, which is not surprising since it needs to determine k for each class, while the best result over different fixed k is selected *a posteriori*. However, X-Means leads to consistently better (significantly so on the Pendigit and Satimage datasets) results than the global classifier, except for the Image dataset. In other words, where improvement with CIC is possible, using X-Means to determine the number of clusters also leads

Dataset	Classes	Dimensions	Training Set Size	Training Class Size	Test Set Size	Test Class Size
Image	7	19	210	30	2100	300
Pendigit	10	16	7494	[719,780]	3498	[335,364]
Satimage	6	36	4435	[415,1072]	2000	[211,470]
Vowel	11	10	528	48	462	42

Table 2. Summary of properties of the benchmark datasets

k	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 8$	$k = 10$	$k = 15$	X
image	92.10	91.71	89.38 ³	89.90 ³	89.71 ³	89.57 ³	90.19 ²	89.81 ³	89.67 ³	89.91 ³
pendigit	95.85	97.48 ³	97.51 ³	98.00³	97.71 ³	97.74 ³	97.88 ³	97.68 ³	97.57 ³	97.97 ³
satimage	86.05	85.90	87.30	88.45 ³	89.35 ³	89.45 ³	89.75 ³	89.60 ³	90.15³	90.00 ³
vowel	51.08	53.25	56.06 ¹	56.71¹	52.16	54.11	51.52	53.68	51.30	53.25

Table 3. SVM results (% accuracy). Note that on 3 of the 4 datasets, CIC-SVM using X-Means to determine k for each class (X column) is significantly better than SVM alone.

k	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 8$	$k = 10$	$k = 15$	X
image	91.67	91.19	88.95 ³	87.33 ³	88.33 ³	88.81 ³	89.62 ²	89.90 ¹	88.95 ³	88.86 ³
pendigit	92.08	95.83 ³	95.71 ³	96.17 ³	96.23 ³	96.54 ³	96.31 ³	96.57³	96.43 ³	96.57³
satimage	83.90	84.35	85.30 ²	86.90 ³	87.90³	87.65 ³	87.45 ³	87.10 ³	87.05 ³	87.25 ³
vowel	46.54	49.13	51.52	46.10	46.32	48.05	45.24	44.59	45.89	48.05

Table 4. BMR results (% accuracy). Note that on 3 of the 4 datasets, CIC-BMR using X-Means to determine k for each class (X column) is significantly better than BMR alone.

k	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 8$	$k = 10$	$k = 15$	X
image	92.71	91.14 ³	90.38 ³	90.67 ³	90.33 ³	90.19 ³	90.19 ³	90.00 ³	89.48 ³	90.42 ³
pendigit	97.80	97.68	97.74	97.74	97.94	97.71	97.88	97.68	97.48	97.77
satimage	91.60	91.55	91.05	90.55 ¹	91.60	91.70	91.65	90.95	90.30	90.65
vowel	66.45	59.31 ³	56.49 ³	56.49 ³	54.98 ³	54.55 ³	51.73 ³	52.38 ³	50.87 ³	51.73 ³

Table 5. RBF SVM results (% accuracy). Using X-Means does not lead to improvements. In general CIC with RBF SVM does not improve on RBF SVM alone.

to an improvement.

3.5 Results with Non-Linear Classifiers

Here we examine performance of CIC using SVM with RBF kernel. Results for different values of k are in Table 5.

The results of the RBF SVM alone on all datasets were higher with any linear classifier, and also somewhat better than most combinations of CIC with linear classifiers. Using CIC (whether with a fixed k or with X-Means) with RBF SVM did not result in any improvements for Image and Vowel datasets, and there was only minor (not statistically significant) improvement on Pendigit and Satimage datasets for intermediate values of k , but not for X-Means. Clearly, non-linear methods such as RBF SVM can represent more complicated decision boundaries - and separate odd-shaped classes better - and therefore stand to benefit less from CIC than linear methods.

These observations support the intuition that CIC improves the performance of linear classifiers by increasing the number of linear decision surfaces and, in effect, approximating with them the non-linear separation boundaries between different classes.

It is possible that, since RBF SVM represent a linear surface in the feature space, clustering in the feature space [8] may allow CIC with RBF SVM to improve over base RBF SVM. However we leave these experiments for future work.

3.6 The Effect of the Training Set Size

Experiments described here aim to show that the CIC results above are not artifacts of the partitions used, and to examine the role of the training set size (both in absolute terms and as a proportion of the dataset).

For each dataset we repeated the following steps 10 times, for values of $f = 0.03, 0.1, 0.3, 0.5, 0.7$:

1. select a fraction f of available data as a training set (maintaining class proportions)
2. train basic classifier, CIC with $k = 2, 3, 4, 5$ on this subset
3. evaluate the resulting classifiers on the remaining points

Results with higher values of k are not shown here because they are not necessary to illustrate the main ideas.

We did not use $f = 0.03$ on the Image dataset because of its small size. We also did not conduct these experiments on the Vowel dataset, because it is small (it has only 90 points for each class making experiments with small f and large f uninterpretable) and because the data has additional internal structure that would be difficult to maintain in order for the results to be comparable to those in the preceding

section (there are 15 distinct speakers, and examples from each should be kept completely either in the training or the test sets).

Note that the Image dataset (with 2310 point and 7 classes) has 330 points per class, Pendigit dataset has approximately 1099 points per class, and Satimage dataset (6435 points and 6 classes) has on average 1073 points per class, with the smallest class of 626 points. This means that the number of points per class is approximately the same for Pendigit and Satimage datasets for the same values of f . For the Image dataset however, with $f = 0.3$ number of points per class is approximately the same as for Pendigit and Satimage with $f = 0.1$; and $f = 0.1$ is comparable to $f = 0.03$ for the larger datasets.

The CIC results, both with SVM and BMR, are given in Figure 2. CIC improves accuracy on Satimage and Pendigit datasets when $f = 0.3, 0.5, 0.7$ of the data is used for training. The improvement is small on Satimage for $f = 0.03, 0.1$, though it is still large on the Pendigit dataset. The best results for both datasets are obtained with $k = 5$. The results on Satimage dataset take a slight dip at $k = 2$ before improving, just as in the previous experiments. CIC outperforms the basic classifier for all f when $k \geq 3$.

For the Image dataset, there is little effect with $f = 0.3, 0.5, 0.7$, and the performance clearly deteriorates when $f = 0.1$. The best results, both with SVM and BMR, are obtained with $k = 2, 3$ for $f = 0.7, 0.5$, and with the basic classifier, $k = 1$, for smaller f .

The plots also demonstrate that with smaller f the results are less stable, as indicated by increases of the standard deviation of the accuracy. This effect is more pronounced on the smaller datasets.

These results confirm observations made previously, that CIC does better than the basic classifier on two datasets and does comparably on another one (Image), as long as the training set is large. Since Image dataset with $f = 0.1, 0.3$ is comparable (in terms of points per class) to Pendigit and Satimage with $f = 0.03, 0.1$, but CIC performs worse than the basic classifier on the Image and not on the other datasets for these values of f , the training set size is not issue. It would appear that for the Image dataset the intrinsic number of components in each class is on average close to 1 (and so CIC gives no improvement), while for the other two datasets it is larger.

4 Discussion

4.1 Related Work

In [2] clustering inside classes is used to speed up training of SVM classifiers. Representatives from each cluster are used to train an initial SVM classifier. This allows approximate identification of potential support vectors for the

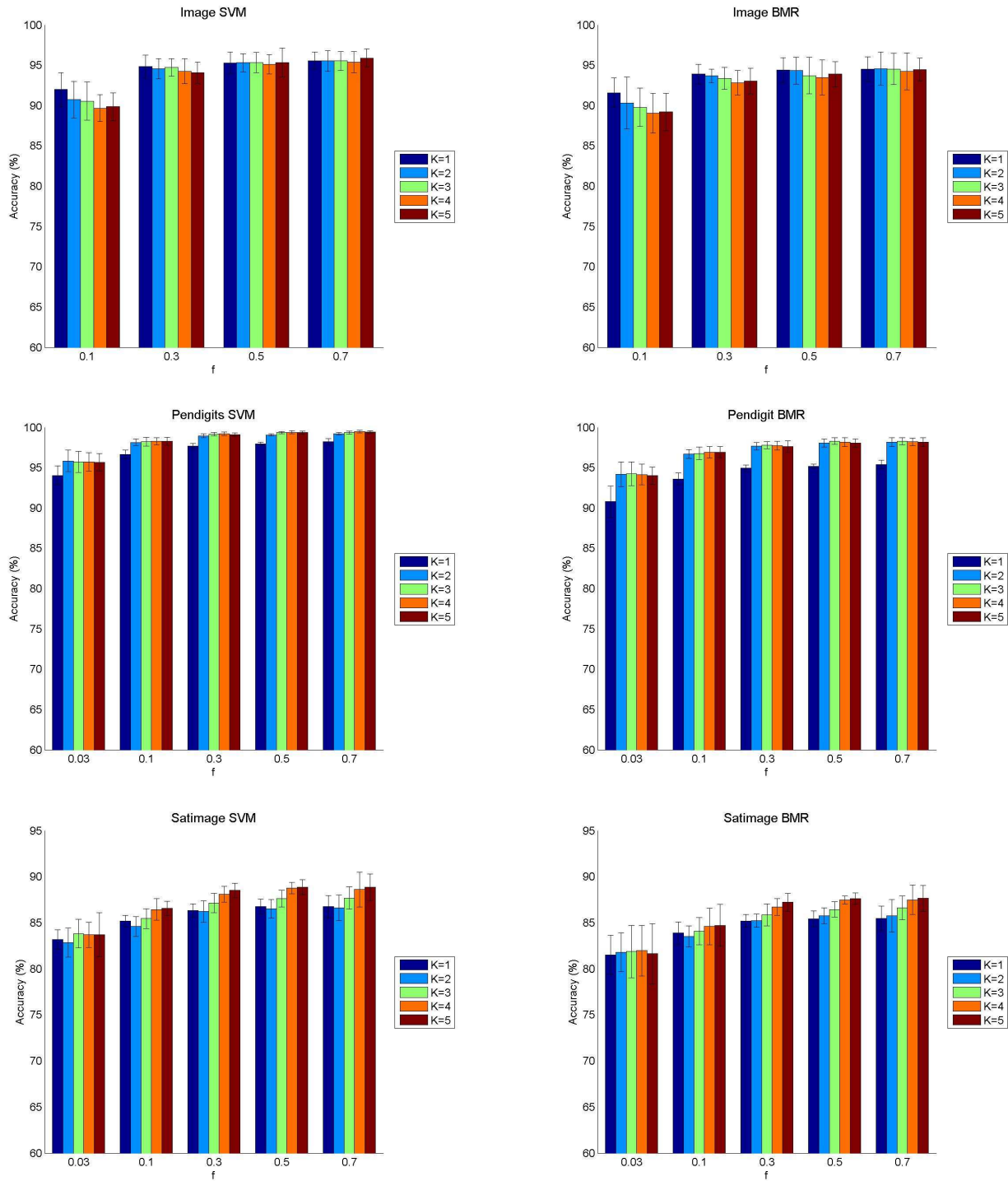


Figure 2. Barplots of % accuracy with linear SVM and BMR for each dataset with $f = 0.03, 0.1, 0.3, 0.5, 0.7$ ($f = 0.01$ is not used on Image dataset) and with $k = 1, 2, 3, 4, 5$. In each plot x-axis shows values of f , while the bars correspond to different values of k . The height of the bar is the mean value over 10 runs of 10-fold cross-validation, while the error-bars are twice the standard deviation.

whole data. Points that are unlikely to be support vectors are ignored. Training on data reduced in this way turns out to be faster than using the whole dataset.

Clustering each class separately is sometimes used in initialization of supervised learning algorithms, such as Generalized Learning Vector Quantization (GLVQ) [20]. GLVQ is a prototype-base classification approach, where each class is associated with a number of prototype (representative) points. A new point is classified by assigning it to the class of the nearest prototype. The prototypes are found by a gradient descent procedure on the training set, starting from some initial setting. The purpose cluster analysis in this approach is to obtain good initial positions for the prototypes.

“Unsupervised output separation” for binary classification was first described in [11], where it was also combined with boosting and bagging:

- During the training stage these steps are repeated 5 times:
 1. Each class is independently partitioned into user-specified number of clusters k ($k = 3, 5$ were the only values considered).
 2. A multi-class classifier (decision tree C5.0 or boosted C5.0) is trained with the cluster labels (rather than with class labels) to distinguish between the clusters.
- In the classification stage, the predictions from classifiers built at each repetition of the training stage are combined. Note that while the classifiers predict one of the cluster labels, these are converted into class labels. In other words, if class 1 has clusters 1,2 and 3 and class -1 has clusters 4,5 and 6, then regardless of which of 1,2 or 3 is predicted by C5.0, the point is assigned to class 1. The two combination methods considered are non-weighted combination, where the number of times a particular class was predicted is counted, and weighted combination, where the probability estimates produced by C5.0 trained on different partitions are added together for each class.

Experiments on 5 datasets from the UCI Repository[1] showed improved classification accuracy on 4 of them. However these improvements were spread out among the combinations of 2 choices of k and the 4 methods tried (i.e. boosted or regular C5.0, and weighted or not-weighted combination). It is unclear from this work whether any single approach consistently leads to improved accuracy. Since all of these approaches involve combining multiple classifiers generated with different clustering results, the improvements could also be due to bagging of such classifiers, rather than to input separation. Finally, since the basic classifiers used in [11] are decision trees, which are capable of

representing complex decision boundaries, the benefits of input partitioning would not be as significant as for linear classifiers. Thus, while this is, to our knowledge, the first paper to introduce a CIC-type approach, it does not provide a clear analysis of the algorithm and of factors leading to improved performance - things that are quite important to any practical work.

In [24] “decomposition of classes via clustering” was analyzed within a framework of Naive Bayes classifier. Improved classification was obtained by addressing “class-dispersion problem” - a situation where a class is spread over several different regions of the feature space. The analysis of the method was specific to Naive Bayes classifiers and not directly applicable to other methods. The use of CIC with other linear classifiers was briefly mentioned in the “future work” section. Clustering was done with Expectation Maximization technique, with the number of clusters in each class chosen via cross-validation.

Recently, cluster analysis inside individual classes was explored in a restricted setting in [13], for balancing class sizes, by splitting the larger class into several clusters. The paper left parameter k to be specified by the user.

4.2 Our Contribution

While the CIC approach seems simple, it is was first discussed, to the best of our knowledge, only in [11]. Since then others [24, 13] have investigated different aspects of such approach, but left gaps that our work aims to fill.

Unlike [11] we obtain a single good clustering result and then build a single multiclass classifier. Therefore, any improvements observed are exclusively due to CIC, and not to bagging or boosting of individual CIC classifiers. [11] also did not address the choice of k , leaving it up to the user. Our results on multiclass problems (as opposed to the two-class problems discussed in [11]), using state-of-the-art classifiers, show that CIC approach improves performance of the linear classifiers. Unlike [24], we examine applicability of the CIC approach to different linear classifiers, not only Naive Bayes, and to non-linear classifiers. We also experiment with an efficient method for automatically selecting k for each class [17], instead of using computationally expensive cross-validationS.

To our knowledge ours is the first work to extensively examine the CIC approach on its own in a general setting (i.e. not tied specifically to problems with class imbalances, or to a particular classifier method or a combination of classifiers). We consider in much greater detail than previous work the effects of the number of clusters, k , and of the training set and class size on the performance of the CIC approach. We also experiment with an *efficient* method for automatically selecting k for each class - an important issue that was not previously addressed.

5 Summary

We have shown that Clustering Inside Classes (CIC) improves accuracy of linear classifiers provided that sufficient training data is available and that the classes have more than one intrinsic component. Even when classes have only one such component, CIC does not cause performance to deteriorate provided there is enough training data for each class.

We examined CIC with different training set and class sizes, using both user-specified and automatically determined values for the number of cluster in a class, k . The choice of k does affect the results, though there tend to be whole ranges of values of k for which CIC improves performance of linear classifiers. We have observed improvements with $10 \geq k \geq 3$ for all but the Image dataset. Using the approach of [17] for automated selection of k improved results compared to a single global classifier. These results were only slightly worse those with the best values of k selected for each dataset *a posteriori*. Prior works did not propose an efficient method for automatic selection of k .

The accuracy improvements obtained with CIC and linear classifiers does not match results of a non-linear classifier. Also, CIC does not improve performance of non-linear classifier. However, CIC allows for other advantages:

- Linear classifiers tend to be faster to train than non-linear ones. A recent paper by Joachims [12] describes a linear time training method for linear SVM. Training several independent linear SVM (which is also easily parallelizable) is thus more efficient than building a non-linear SVM.
- Linear classifiers tend to be faster at classification time. In order to apply a linear SVM to a new case only one similarity (inner product) computation is needed. Therefore, only k such computations are needed to apply CIC. However, applying non-linear SVM requires computing similarity with each support vector, of which there are frequently hundreds or thousands.
- Linear classifiers are much more interpretable. Their weights indicate which features are important for classification and how they affect predictions. In many applications being able to understand how classifier makes predictions is almost just as important as having a high accuracy.

An interesting avenue for further work is to combine CIC with non-linear classifiers by performing cluster analysis in a non-linear space. A number of techniques for performing such analysis have been described, for example in [8].

The datasets examined in our work arose in signal-processing applications. Text and bioinformatics datasets are frequently characterized by large number of features, sometimes exceeding the number of available training

points. While we believe that the CIC approach will also be successful in such applications, the characteristics of the data would have to be taken into account, most likely by utilizing a more appropriate clustering approach.

Another direction for future investigations is development of visualization/representation methods for the extracted class structure (i.e. within-class clusters) as a data exploration method.

The results described here demonstrate usefulness of and provide insights into the CIC approach and encourage further investigations and applications.

6 Acknowledgments

The author would like to thank Casimir Kulikowski and Ilya Muchnik for many helpful discussions, and Fabian Moerchen for valuable comments.

References

- [1] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.
- [2] D. Boley and D. Cao. Training support vector machine using adaptive clustering. In *Proceedings of SDM'04*, 2004.
- [3] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.
- [4] T. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1985–1923, 1998.
- [5] C. Farley and A. E. Raftery. Model-based clustering, discriminant analysis and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, June 2002.
- [6] E. Forgy. Cluster analysis of multivariate data: efficiency vs interpretability of classification. *Biometrics*, 21:768, 1965.
- [7] D. Fradkin. *Within-Class and Unsupervised Clustering Improve Accuracy and Extract Local Structure for Supervised Classification*. PhD thesis, Rutgers, The State University of New Jersey, January 2006.
- [8] M. Girolami. Mercer kernel based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784, 2002.
- [9] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2), March 2002.
- [10] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.
- [11] N. Japkowicz. Supervised learning with unsupervised output separation. In *Proceedings of the IASTED ASC'2002*, pages 321–325, 2002.
- [12] T. Joachims. Training linear svms in linear time. In *Proceedings of KDD'06*, pages 149–158, 2006.
- [13] P. W. Junjie Wu, Hui Xiong and J. Chen. Local decomposition for rare class analysis. In *Proceedings of the KDD 2007*, pages 814–823, 2007.

- [14] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:128–137, 1982.
- [15] D. Madigan, A. Genkin, D. D. Lewis, S. Argamon, D. Fradkin, and L. Ye. Author identification on the large scale. In *Proceedings of Joint Annual Meeting of the Classification Society of North America*, 2005. Software: www.bayesianregression.org/bmr.html.
- [16] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 58(2):159–179, 1985.
- [17] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the ICML'00*, pages 727–734, 2000.
- [18] A. Rida, A. Labbi, and C. Pellegrini. Local expert combination through density decomposition. In *Proceedings of the IEEE International workshop on AI and Statistics*, pages 130–136, 1999.
- [19] S. Salvador and P. Chan. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *Proceedings of the ICTAI'04*, pages 576–584, 2004.
- [20] A. Sato and K. Yamada. Generalized learning vector quantization. In *Proceedings of ICPR'98*, 1998.
- [21] B. Tang, M. I. Heywood, and M. Shepherd. Input partitioning to mixture of experts. In *Proceedings of IEEE World Congress on Computational Intelligence (IEEE WCCI 2002)*, 2002.
- [22] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.
- [23] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 2nd edition, 1998.
- [24] R. Vilalta and I. Rish. A decomposition of classes via clustering to explain and improve naive bayes. In *Proceedings of the 14th European Conference on Machine Learning (ECML 2003)*, pages 444–455, 2003.
- [25] S. M. Weiss and C. A. Kulikowski. *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann Publishers Inc., 1991.
- [26] L. Xu, M. I. Jordan, and G. E. Hinton. An alternative model for mixtures of experts. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 7, pages 633–640. 1995.