DIMACS Technical Report 2004-02
April 2004

A Study of K-Means Clustering for Improving Classification Accuracy of Multi-Class SVM

by

Dmitriy Fradkin
Dept. of Computer Science
Rutgers University
New Brunswick, New Jersey 08854

Ilya Muchnik
DIMACS
Rutgers University
New Brunswick, New Jersey 08854

ABSTRACT

This work discusses how clustering methods, in particular K-Means, can be used to improve classification accuracy. We discuss two approaches to constructing hierarchical classifiers using cluster analysis and suggest new methods and improvements in each of these approaches. We also suggest a new method for constructing features that improve classification accuracy. All the methods are evaluated in the context of multi-class classification problems using multi-class SVM and some standard datasets from UCI.

1 Overview

We discuss the uses of cluster analysis in supervised learning and, more specifically, in building hierarchical classifiers.

In the literature we have found two approaches. One approach can be seen as "geometric" in that it partitions (possibly in a fuzzy way) objects into a number of groups based on their closeness or similarity, and then constructs a classifier for each of the groups. The predictions of lower-level classifiers are then combined together to make a final prediction. This is the most frequent approach to building hierarchical classifiers [12, 5, 6]. The other approach is in a sense "statistical" and has been applied to multi-class problems. Here the classes are grouped into "metaclasses" based on the cluster analysis of their confusion matrix, i.e. on the frequency of misclassification of points between different classes. A meta-classifier is built to distinguish groups of classes, and lower-level classifiers are used to separate individual classes inside such a metaclass [9].

The central idea in both of these approaches is that a problem being solved at the higher level (separating clusters or metaclasses) is rather easy, compared to the original problem. The problems of separating classes at the lower level are more difficult than at the higher level, but obtaining the solution is aided by the fact that these problems are smaller and more local than the original problem. Hierarchical classifiers rely on the idea that the errors accumulated at the two levels will be fewer than with a global single-level approach.

We suggest new methods and directions for both of these approaches. Furthermore, the effectiveness of our methods has led us to consider an altogether different direction of work. If clustering data into groups leads to construction of smaller and easier problems, then constructing features that are indicative of the relation of the objects to these groups should also improve classification. Our experiments confirm this is indeed the case. While a similar method has been developed specifically for textual data [4], our approach is more general and its assumptions are not tied to any particular type of data.

In the existing literature, especially that on "geometric" hierarchical classifiers, fuzzy clustering methods or mixture models are frequently used. These methods are computationally expensive. Their performance is difficult to evaluate from the point of view of supervised learning since they produce sets of weights or density models that can have significant influence on the performance of a classifier where they are used. A slight change to these parameters may produce a different classification rule. A hard partitioning method, such as K-Means clustering, has the advantage of being efficient and providing simple output, making it easy to evaluate on the basis of supervised learning results. We systematically examine effectiveness of K-Means in the context of the above-mentioned approaches.

Additionally, the use of K-Means in supervised learning is relatively unexplored. On the one hand, more sophisticated methods are often suggested without providing comparison to K-Means. On another hand, as this work demonstrates, many significant modifications to existing methods and new approaches to problems can be suggested. Multi-class classification problems, occurring in many areas of human activity, constitute an especially fertile ground for developing and applying new ideas.

In order to make the work manageable, we restrict ourselves to using popular and powerful SVM classifiers [14]. Since we evaluate all of our methods on multiclass problems, we use a particular version of SVM developed specifically for such situation [3].

This report is structured as follows. We first introduce our notation and some background material necessary for our work. We then briefly talk about datasets and feature normalization method used in our experiments. Then, in Sections 4-6, for each of the three approaches we discuss existing literature, suggest improvements or alternative methods, and describe our experimental results and future work. We conclude by comparing, in general terms, effectiveness of these three approaches and make some suggestions on how they can be applied to new data.

2 Background

2.1 Notation

Before proceeding we would like to specify our notation. Symbols are described in Table 1.

We describe some of the more important symbols in more detail here. Capital case K is the number of classes in a training set W, and is therefore a constant. Classes are denoted by labels $L_1, \ldots, L_K$. Lower case k denotes the number of clusters and is usually an argument to K-Means procedure. In a number of methods discussed here, the value of k is also important for constructing classifiers.

Letter R is used to denote SVM classifiers. Essentially, a classifier is a function, that, given a point x, returns a label. R_0 will be used to denote a global classifier. It always returns a label belonging to the set of class labels of W: $R_0(x) \in L_1, \ldots, L_K$. In sections that discuss hierarchical classifiers, symbol R_1 will be used to denote first level classifier. The labels returned by this classifiers do not necessarily correspond to the class labels. Another notation is R_{2i} - ith classifier on second level.

Throughout this paper we use the words "vector" and "point" interchangeably, depending on which is more suitable in the context of the discussion. Therefore, unless otherwise specified, phrases "point x" and "vector x" refer to the same object, in the same Euclidean space.

2.2 K-Means

A classical clustering procedure K-Means, given a parameter k, partitions a training set W into k clusters: $W = \bigcup\limits_{i=1,\ldots,k} S_i$, $S_i \bigcap S_j = \emptyset$, $\forall i \neq j$. K-Means attempts to find a partition that minimizes the following criterion of quality:

$$E = \sum_{i=1}^{k} \sum_{x \in S_i} ||x - s_i||^2, \tag{1}$$

Symbol	Meaning
W	Dataset, i.e. all points
$N = \|W\|$	Size of the dataset
K, k	Number of classes, clusters in W
$L_1, \ldots, L_K$	Classes of W
S, S_i	A cluster in W: a set of points of W
s, s_i	center of cluster S, S_i
d	dimensionality of the data
$x, y \in X$	data points, belonging to a space X
$x \cdot y$	inner product between two vectors
l_j	label of the jth point
$l_j(x)$	an indicator function of class j
$\rho(x, y, \alpha)$	a radial basis similarity function with parameter α
$M(x)$	a mapping of point x into a new space
$\|\|x\|\|$	L_2 norm of vector x
w, w_h	a weight vector (of class h)
ξ	vector of slack variables
ξ_j	value of slack variable for the jth point
b	bias
$R_{ij}(x)$	SVM j at level i of hierarchy, returns label of x
$nsv(R_{ij})$	number of support vectors used by R_{ij}
A	An $n \times n$ confusion matrix
m	Number of meta-classes or meta-clusters

Table 1: Notation

where k is the number of clusters, $s_i = \frac{\sum\limits_{x \in S_i} x}{|S_i|}$ (i.e. s_i is the center of cluster S_i). This criterion was first described by Lloyd [11] and Fisher [7] and had since been used in many applications.

2.3 Multi-class SVM

Given N training examples $(x_1, l_1), \ldots, (x_N, l_N)$, the problem of constructing a linear SVM classifier for a two class problem (positive and negative classes), can be formulated as the following optimization problem [14]:

$$\min_{w,\xi} \frac{1}{2}||w||^2 + C \sum_{j=1}^{N} \xi_j, \tag{2}$$

subject to constraints:

$$\forall j = \overline{1, N}: \ \xi_j \geq 0, \ l_j \cdot (w^T \cdot x_j + b) \geq 1 - \xi_j, \tag{3}$$

where $l_j \in \{-1, +1\}$ depending on whether x_j belongs to the negative or to the positive class.

Cramer and Singer [3] suggested the following approach to multi-class problems:

$$\min_{w_1,\ldots,w_K,\xi} \frac{1}{2} \sum_{h=1}^{K} ||w_h||^2 + C \sum_{j=1}^{N} \xi_j, \tag{4}$$

subject to:

$$\begin{aligned} \forall j = \overline{1, N}, \ h = \overline{1, K} : \xi_j \geq 0, \ h \neq l_j, \\ w_{l_j}^T \cdot x_j - w_h^T \cdot x_j \geq 1 - \xi_j. \end{aligned} \tag{5}$$

The resulting decision function is:

$$l_j(x) = \begin{cases} 1 & \text{if } j = \text{argmax}_{h=1,\ldots,K}(w_h^T \cdot x) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

In all of our experiments, we use multi-class linear SVM mode of BSVM [2] (parameter settings "-s 2 -t 0"), that implements the above method.

3 Data

3.1 Datasets

For the experiments we used four well-known datasets from UCI [1].

1. Image Segmentation dataset has 7 classes, 19 features and 2310 observations (210 train and 2100 test), evenly divided between classes. The points correspond to 3x3 pixel regions randomly drawn from 7 outdoor images (with the name of the image serving as a label). The features are computed based on color intensities and coordinates of the pixels in the regions. All features are numeric.

2. Pendigit dataset consists of 10992 points represented by 16 features and a label (a digit from 0 to 9). This dataset was created by collecting 250 samples of digits written on a pressure-sensitive tablet from each of 44 writers. The features were obtained by spatial resampling of the points in the plane and are integers in the range [0,100]. Training set has 7494 points, while test set has 3498 points.

3. Satellite Image (Satimage) consists of 6 classes, 36 features and 6435 (4435 train and 2000 test) observations. Points correspond to multi-spectral values of pixels in a 3x3 regions of a satellite image (4 spectral bands for each of the 9 pixels). The features are integers, in the range [0,255]. The labels are soil types.

4. The fourth dataset was a **Vowel dataset**. There are 990 points, corresponding to 11 English vowel sounds, represented by 10 features and a label. The features are derived from analysis of sample windowed segments of the speech signal and are real-valued. The training set consists of 528 points, and the test set has 462 points. The classes are represented approximately evenly in the training and test sets.

3.2 Feature Normalization

In machine learning, data is frequently preprocessed. This is done to address possible problems with features.

The scales of individual features can differ drastically. Such disparity is often caused by use of specific units of measurement and is not an inherent characteristic of the data. However, this may present problems for many machine learning methods since features with different scales require very different weights. Another possibility is that, while the features may have the same approximate scale, the distribution of their values has different means and variance. This may lead to the same problems as having features on different scales.

There are many methods for addressing these issues. The choice of a particular method may have a great effect on the performance of the machine learning methods. The best method is frequently problem-specific. Since exploration of different normalization methods is not the subject of this work, we decided to use a popular technique known as statistical normalization. Each feature is independently transformed to have 0 mean and variance of 1:

$$x_{ij} = \frac{x_{ij} - \overline{x_j}}{\sigma_j} \tag{7}$$

where $\overline{x_j}$ is the mean value of jth coordinate and σ_j is its standard deviation, estimated on the training set. The normalization is applied both to the training and the test sets, using $\overline{x_j}$ and σ_j computed on the training set. We conducted all the experiments on data that underwent feature normalization according to (7).

4 “Geometric” Hierarchical Classifiers

4.1 Introduction

Most hierarchical approaches (also known as ensembles of classifiers, local experts, etc.) involve partitioning training data and training separate classifiers on different parts of training data. The intuition behind this is that use of several “local” classifiers (classifiers that are trained on points in a small part of the input space) will result in simpler decision rules and better performance than would be seen from a globally trained expert.

Once local classifiers are built, classification of a new point proceeds as follows. A new point is first assigned to one or more clusters, and classified separately by classifiers trained on these clusters. The results from different classifiers are then combined using majority vote or some form of weighted voting, where a vote of local classifier is weighted by the distance of a point to that classifier’s cluster. Examples of voting rules used in this context are given in [12], [6].

We suggest a new architecture for hierarchical methods. Part of the novelty lies in using SVM at the first level, to assign points to regions. To the best of our knowledge, in the literature, this is usually done based on distance to cluster centers or on cluster density at a point. Additionally, one of the methods that we suggest is capable of choosing whether to use local classifier or a global global one based on the quality of the region and/or of the local decision rule. More specifically, this method estimates the quality of classifier in a region based on number of points and number of support vectors used. If the expected quality is poor (the fraction of support vectors in the region is high) than a global classifier is used instead of a local one. Additionally, all of our experiments are done using K-Means clustering, which is shown to yield good classification results. This method is simpler than fuzzy clustering methods frequently used in the literature.

In the following section we discuss several recent papers on hierarchical classification. Section 4.3 describes in detail our two methods. We then discuss experiments and results, followed by conclusions and directions for further work.

4.2 Existing Work

A number of recent papers examined hierarchical approach to classification.

Fuzzy clustering via Mixtures of Gaussians was used to partition data into k clusters (with some degree of overlap) in [12].The authors experimented both with MLP (Multi-Layer Perceptrons) and SVM (Support Vector Machines) as local classifiers.

This paper assumes that the input density is a mixture of Gaussian densities and estimates their parameters from the data. The data is initially clustered using Kohonen map with conscience. The number of resulting clusters k is taken to be the number of components in the mixture. Once this is determined, the mean m_j and covariance matrix Σ_j are estimated for each cluster j. The local input density function is assumed to have the following form:

$$g_j(x) = \frac{1}{(2\pi)^{n/2}|\Sigma_j|^{\frac{1}{2}}} exp(-\frac{1}{2}(x - m_j)'\Sigma_j{}^{-1}(x - m_j)) \tag{8}$$

The global input density is:

$$g(x) = \sum_{j=1,\ldots,k} \mu_j g_j(x),\ \mu_j \geq 0,\ \sum_{j=1,\ldots,k} \mu_j = 0. \tag{9}$$

Each point x in the training set is assigned to all clusters j such that $g_j(x) > \Theta$, where Θ is a threshold parameter. A local classifier $f_j(x)$ is trained on jth group points. Local classifiers are combined in the following way:

$$F(x) = \frac{1}{\sum\limits_{j=1,\ldots,k} \alpha_j g_j(x)} \sum_{i=1,\ldots,k} \alpha_i g_i(x) f_i(x). \tag{10}$$

Parameters α_i can be chosen in a number of different ways. Setting $\alpha_i = \mu_i$ leads to weighted averaging. A different combination method, called adaptive, adjusts α_i to minimize some error, defined for a global rule on a hold-out validation set.

Performance of hierarchical methods was compared with that of a single global classifier and of combination of global classifiers trained on different subdivisions of the data. Experimental results suggest that hierarchical classifiers may perform much better than global classifiers.

In [5], Generalized Learning Vector Quantization (GLVQ) is used to create a prototype vectors that induce a partition of the space into k clusters, where k is a parameter. MLP is used as the local classifier. For training of local classifiers, each training point is replicated into the regions associated with j nearest prototypes.

A new point is assigned to the j nearest prototypes and classified with corresponding classifiers. The outputs of these MLP are averaged together to produce a decision. Experiments on several lowercase character database suggest that this method outperforms Nearest Neighbor methods, MLP and boosted MLP trained on the whole input space. It is noted that the choice of the number of prototypes has a large influence on the accuracy of the classifier.

Several methods for partitioning the training set followed by training a decision tree (C4.5) classifier on each set in the partition are examined in [6].

The first partition method discussed is hard clustering via K-Means. This method results in disjoint partitions of the training set. A different classifier is trained on each cluster in the partition. When a new point comes in, it is assigned to the cluster with the nearest centroid and is classified using the classifier trained on that cluster.

Another method is called by the authors "soft replication." Given a a minimum distance from a training example x to the closest center s_i: $||x - s_i||$, example is replicated into to all clusters whose center s_j lies within $\alpha||x - s_i||$. Here α is a pre-specified parameter. This leads to an overlapping groups of training points, with each point belonging to at least one of the groups. A classifier is trained on each group. A test point x is assigned to all clusters with centers that are closer than $\alpha||x - s_i||$ to it. The votes of corresponding jth classifier is incorporated with weight $||x - s_i||/||x - s_j||$.

"Fuzzy replication" is a fuzzy membership-based replication strategy. The membership of a point x in a cluster S_j is calculated as:

$$u'_j(x) = \Big(\sum_{i=1,\dots,k} \Big(\frac{||x - s_j||}{||x - s_i||}\Big)^2\Big)^{-1}. \tag{11}$$

and are then constrained to sum to 1 for each x. Let $u_j(x)$ be equal to $u'_j(x)$ after rescaling. Let $u_m(x)$ be the maximum membership value for x. A training point x is replicated into clusters for which its membership exceeds a threshold T, specified as $T = u_m(x) - \epsilon$, where ϵ is a free parameter (that was set to 0.2 in experiments). Classifiers are then built based on such overlapping partitions (disregarding the weight of points in the clusters). For classification, membership values of a new point in each cluster are computed. Where these values exceed the threshold T (the same one that used for replication), the classifiers for corresponding clusters are queried to classify the test point. Their predictions are combined by weighted voting, where the weights are the membership values of the test point for the corresponding clusters.

"Possibilistic replication" is a soft clustering and replication approach similar to fuzzy clustering but where the membership values are not constrained to sum up to 1. In this paper they were computed as:

$$u_j(x) = \frac{1}{1 + ||x - s_j||^2}. \tag{12}$$

Membership values that are greater than $\epsilon u_m(x)$ indicate that replication should take place. In these experiments, ϵ was taken to be 0.5. Classifiers are trained on each group of points. A new point is classified by a weighted vote of classifiers whose cluster centers are closer than ϵu_{max}. The weights are the membership values u_j, $j = 1, \dots, k$.

The above four methods were compared against each other and against decision tree (C4.5) trained on the whole training set on 6 datasets from UCI Machine Learning Repository. The number of clusters K was varied from 2 to 26 for all datasets and methods.

For most values of K hard partitioning method performed worse than the plain decision tree. Comparison between the remaining three strategies does not reveal a clear winner. However both soft and possibilistic methods are ranked highest much more often than fuzzy method. The significant result is that the best results achieved on each dataset in the experiments were obtained with soft or possibilistic methods (and not with a decision tree trained on the whole dataset).

This short review is meant to give the reader an idea of both common trends in hierarchical algorithms, and of great variety of methods used.

4.3 Our Approach

In the literature we found an inconsistency. Most hierarchical classification papers use a number of fuzzy clustering methods. Only one paper actually compared K-Means with fuzzy clustering [6]. They claimed that resulting classifier performs worse than a global one. Our results demonstrate that this need not be the case.

We experimented with two hierarchical architectures, using hard clustering in both cases. The first of these can be called Simple Hierarchy (SH) because it straightforwardly creates a partition and uses a single local classifier for each test point. The steps for constructing decision rule are described in Algorithm 1. A new point can then be classified using Algorithm 2.

Algorithm 1 SH Training Algorithm

Require: A set W, number of clusters $k > 0$.
Partition W into k clusters using K-Means procedure
Train a global multi-class SVM to distinguish between the k clusters: R_1
for $i = 1, \ldots, k$ **do**
    train a local multi-class SVM to recognize only classes present in the cluster S_i: R_{2i}
end for

Algorithm 2 SH Classification Algorithm

Require: A point x.
Let $i = R_1(x)$, $i = \overline{1,k}$ - cluster to which x is assigned.
Return $R_{2i}(x)$

The SH approach may suffer from lack of training data in a region. When the number of points in a cluster is small, the resulting classifier will not be very reliable. Even when a region has a lot of points, if they belong to many different classes, we may fail to obtain a good decision rule.

We can detect clusters that do not have a good decision rule by considering the number of support vectors used by a local classifier relative to the total number of points in the cluster. Let us denote this ratio for a cluster S_i and a classifier R_{2i} trained on this cluster by q_i:

$$q_i = \frac{nsv(R_{2i})}{|S_i|} \tag{13}$$

If q_i is higher then some threshold, γ, then there are likely to be many errors. (According to [14], the number of support vectors provides an upper bound on the training error). If that is the case, we might be better off using a global classifier rather than the local one. These considerations lead to Hierarchy with Global Classifier (HGC). Training and classification using this approach are described in Algorithms 3 and 4.

We note that if $\forall i = \overline{1,k}\ \gamma > q_{2i}$, then HCG reduces to SH. However, SH and HCG may produce identical labels even if that is not the case.

Algorithm 3 HGC Training Algorithm

Require: A set W, number of cluster $k > 0$.

Train a global classifier R_0
Partition W into k clusters using K-Means procedure
Train a global multi-class SVM to distinguish between the k clusters: R_1
for $i = 1, \dots, k$ **do**
    train a local multi-class SVM to recognize only classes present in the cluster S_i: R_{2i}
    Compute indicator of quality of R_{2i}: q_i, according to (13).
end for

Algorithm 4 HGC Classification Algorithm

Require: A point x, threshold $0 \le \gamma \le 1$.

Let $i = R_1(x)$, $i = \overline{1, k}$.
if $q_i > \gamma$ **then**
    Let $j = R_0(x)$.
else
    Let $j = R_{2i}(x)$.
end if
Return j.

4.4 Experiments

The classifier that we used for all experiments is multi-class linear SVM mode of BSVM [2] (parameter settings "-s 2 -t 0"). For each dataset, we experimented with both SH and HGC using $k = 5, 10, 20$. Parameter γ was set to 0.5 in all HGC experiments. The same first level classifier R_1 was used for SH and HGC in each experiment.

Results of experiments are presented in Tables 2-5. The first row of each contains header, and the remaining rows contain results for $k = 5, 10, 20$. Second and thirds column contain accuracies and number of support vectors of a global classifier. The two columns called SH and HGC contain accuracies of the corresponding methods. The last two columns show respectively the number of support vectors at the first level (same for both SH and HGC), and the average number of support vectors per cluster on a second level (for SH).

k	Original	$nsv(R_0)$	SH	HGC	$nsv(R_1)$	$\frac{nsv(R_{2*})}{k}$
5	90.95	68	91.33	91.43	27	11.8
10	90.95	68	87.62	90.71	68	6.8
20	90.95	68	87.43	89.29	90	3

Table 2: Image Dataset, Hierarchical Classification Results

k	Original	$nsv(R_0)$	SH	HGC	$nsv(R_1)$	$\frac{nsv(R_{2*})}{k}$
5	90.85	1016	95.25	95.25	223	92.8
10	90.85	1016	96.05	96.05	499	44.4
20	90.85	1016	96.97	96.97	755	22.55

Table 3: Pendigit Dataset, Hierarchical Classification Results

k	Original	$nsv(R_0)$	SH	HGC	$nsv(R_1)$	$\frac{nsv(R_{2*})}{k}$
5	82.6	1409	82.7	82.1	1729	251.2
10	82.6	1409	85.05	83.4	1158	122.2
20	82.6	1409	85.85	82.7	1988	58.1

Table 4: Satimage Dataset, Hierarchical Classification Results

4.5 Discussion of Results

Table 2 contains results of experiments on the Image dataset. Here the results of a global classifier are quite good. However both SH and HGC give better results with $k = 5$ (91.33 and 91.43 respectively). Also, HGC always gives better results than SH. Results deteriorate with increase in number of clusters, and are worse than that of global classifier for $k = 10, 20$. This is likely due to the small size of training set, which results in small clusters with insufficient amount of training data, as k increases. For $k = 5$ cluster sizes are 6, 30, 35, 56 and 83. However for $k = 20$, eight clusters have 3 or fewer points, while the largest cluster has 30.

Results for Pendigits dataset are in Table 3. The results of SH and HGC are identical for all values of k and are much better than those of a global classifier (95.25 - 96.97 vs. 90.85). Examination of q_{ij} values showed that the number of support vectors in all clusters was much lower that $\gamma = 0.5$, causing SH and HGC to produce identical results. We also note that here, as opposed to Image Dataset, increasing k led to better results. Since this dataset has a large training set (7494 points), it supports our view that in Image Dataset cluster size becomes too small as k increases.

Results for Satimage dataset are presented in Table 4. Here HGC leads to performance comparable to that of a global classifier (82.1, 82.7 and 83.4 for HGC vs 82.6 for R_0), while SH produces somewhat better results (82.7 - 85.85). Additionally, while SH scores improve as k increases, this is not the case for HGC, where the highest score is for $k = 10$.

Table 5 contains results of experiments on the Vowel dataset. Here the results with the global classifier are rather poor (37.45). Both hierarchical methods perform better. SH is

k	Original	$nsv(R_0)$	SH	HGC	$nsv(R_1)$	$\frac{nsv(R_{2*})}{k}$
5	37.45	456	43.51	43.29	66	56.8
10	37.45	456	38.91	38.96	104	24.2
20	37.45	456	46.10	44.59	157	11.75

Table 5: Vowel Dataset, Hierarchical Classification Results

best for $k = 5, 20$ (43.51 and 46.1), while HGC is best for $k = 10$ (38.96). Overall, SH has better results. The relation between k and accuracy is unclear.

It is clear that the number of support vectors on the first level increases proportionately to k (for all datasets except for Satimage), while the average number of support vectors at second level decreased at approximately the same rate. This makes sense, since increasing number of clusters puts greater emphasis on the first level classifier and simplifies task for lower level classifiers.

For Vowel and Pendigit datasets, the average number of support vectors used by SH (i.e. the sum of support vectors on first level with the average number of support vectors on second level) was always significantly less than that of a global SVM. For Vowel the numbers for SH vary from $66 + 56.8 = 122.8$ to $157 + 11.75 = 168.75$ compared with 456 for R_0. For Pendigit, the numbers are in a range of 300 to 800 compared to 1016 for R_0. For Satimage and Image datasets this wasn't always the case. For example for Image dataset, $k = 20$ results in average of $90 + 3 = 93$ support vectors versus 68 used by R_0. Note however, that when the average number of support vectors in SH is less than that in R_0, SH outperformes the global classifier.

Overall, out of 12 comparisons (3 per dataset), SH performed better than HGC 5 times, worse than HGC - 4 times, and identically - 3 times. On the basis of these results we cannot determine if one of these methods is better than the other.

4.6 Conclusions and Future Work

Our experiments suggest that the proposed two schemes for hierarchical classification perform better than a global classifier. It is not clear from the results whether one of the methods is better than another.

The choice of number of clusters, k, considered with respect to the total number of points, has significant effect on the classification accuracy.

Our work may be extended by considering different variants of HGC, for example by taking into consideration the number of classes in a cluster, to decide whether a local or a global classifier should be used. Different values of γ or methods for choosing it automatically should also be considered.

5 Confusion Matrix Analysis

5.1 Introduction

An important group of supervised learning problems are multi-class problems, that is problems where it is necessary to distinguish between several classes of data.

Building classifiers for such problems has its own difficulties. Usually, distinguishing between several classes is more difficult than distinguishing between only two. Training data from all classes is necessary and the decision rule will necessarily have to be more complicated.

Let us assume however that somehow a multi-class classifier R_0 for K classes is built and then tested (for example using cross-validation or a hold-out set). The result of testing is usually a confusion table, an $K \times K$ matrix A where entry a_{ij} contains a fraction (or exact number) of points of class i in validation set that were classified as belonging to class j.

For a two class problem, such table tells us is the accuracy of our classifier on each of the two classes. However, in the case of a multi-class problem we have much more information available. Assuming that the classifier was tested on a representative sample of the data space, we know exactly what classes, and to what degree, are confused by the classifier we built.

Should all classes be confused with all other classes at approximately the same rate, this information is not of much use. However, it is reasonable to consider the situation (which often occurs in practice [9]), where each class is confused only with few other classes, or that all classes are recognized well except for a few that are confused with all others. In other words, it is reasonable to expect confusion matrix to have some structure that reflects relations between the classes. If we could combine classes that are confused with each other into a small number of groups ("metaclasses"), it is possible that our classifier R_0 would make no or few mistakes distinguishing between these metaclasses and almost all mistakes would be inside them.

Thus, while our classifier possibly makes a lot of mistakes distinguishing between all the classes, it can be very good in distinguishing between metaclasses. Now it makes sense to try to build classifiers that would distinguish only between the classes belonging to the same metaclass. While this requires additional effort, each new classifier is likely to be better than our initial, more complex classifier for all classes, even though it uses less data to train on. We effectively split our problem into several smaller ones, in a hierarchical fashion. The way we would classify a new example is first to assign it a label l using initial classifier R_0. Then we classify it using a classifier built to distinguish the classes inside a metaclass containing class with label l.

We thus arrive at a two-level architecture for classification. It remains to specify a method for grouping classes based on confusion matrix.

5.2 Existing Work

Such approach has been discussed in [9]. The authors work with text classification problems. They use confusion matrix analysis to create a two-level hierarchical classifier. In one case, they train Naive Bayes (NB) classifier on all classes and obtain a confusion matrix of its performance on a validation set. Representing each class by normalized version of corresponding row in confusion matrix A and using Euclidean distance, they perform agglomerative clustering (using Ward's method) and obtain a tree of classes. This tree is clipped at a point where the number of clusters corresponds to the inflection point on a graph of cluster merge distances [8]. The resulting partition gives the desired metaclasses. Then, powerful SVM classifier is trained to distinguish between metaclasses. Also, lower level SVM classifiers are trained to separate classes inside each metaclass.

The authors also explore several other two-level combinations (NB on both levels, NB on first level and SVM on the second one) and compare them with global NB and SVM classifiers. Experiments are also done to determine the accuracy on each level separately (for the second level this refers to accuracy on the data classified correctly by the first level). While the accuracy at each level of the two-level architecture (with SVM on second level) was higher than for any of the global classifiers, the compounding of errors led to worse results than for the global "one-against-all" SVM approach (which showed the best accuracy among all examined classifiers).

We will refer to the approach used in [9] as ReTraining First Level (RTFL) approach, since a new first level classifier is trained to distinguish metaclasses. The algorithms for training classifiers and labeling new data are given as Algorithms 5 and 6 respectively.

Algorithm 5 Classifier Training with Confusion Matrix Analysis: RTFL Method

Require: A set W with K classes.

1: Train a global classifier R_0 to recognize K classes
2: Use hold-out set or cross-validation on W to obtain confusion matrix A.
3: Partition classes based on A in m groups, to minimize the number of mistakes outside of groups
4: Train first level classifier R_1 to distinguish between m groups (metaclasses)
5: **for** $i = 1, \ldots, m$ **do**
6: Train R_{2i} to separate classes that fall into group i
7: **end for**

Algorithm 6 Classification with Confusion Matrix Analysis: RTFL Method

Require: A point x.

1: Let $i = R_1(x)$, $i = \overline{1, k \cdot K}$: i is a metaclass
2: Return $R_{2i}(x)$.

We would like to make note of several factors. The goal of constructing metaclasses is to find groups with fewer errors between them. Since the metaclasses are built using confusion table generated by a classifier R_0, it is important for the first-level classifier in the final construction to make mistakes exactly as R_0 would, so that the errors still fall into the same metaclass and can be corrected on the second level. But this can be achieved simpy by using R_0 as a first-level classifier - there is no need to train new classifier to distinguish between metaclasses. The fact that this reasoning was not followed in the above approach ([9]) may be responsible for its unsatisfactory performance.

Another comment concerns the method used to derive meta-classes. Treating rows of A as vectors representing classes takes into consideration only on-way errors, i.e. row representing L_i contains the points misclassified as L_j (in cell a_{ij}), but no information on how many points of L_j were misclassified as L_i. Values a_{ij} and a_{ji} need to be combined to obtain similarity between classes L_i and L_j. Also, minimizing distances between rows in a cluster does not directly minimize the number of errors between metaclasses. A method for computing metaclasses that would account for these points is desirable, and suggested furthe below.

5.3 Our Approach

Here we suggest some variations or improvements of ideas presented in [9] that use traditional clustering methods.

5.3.1 Cluster Analysis of Confusion Matrix

Given a $K \times K$ confusion matrix $A = ||a_{ij}||$, let B be a symmetric matrix, $||b_{ij}|| = B = \frac{1}{2}(A + A^t) = ||\frac{1}{2}(a_{ij} + a_{ji})||$.

We view matrix B as describing similarities between classes: $b_{ij} = b_{ji}$ is similarity between classes i and j. This suggests that a good partition of B should aim to maximize the sum of intra-cluster similarities:

$$E_1 = \sum_{u=1}^{k} \frac{1}{|S_u|} \sum_{i,j \in S_u} b_{ij}. \tag{14}$$

We consider the criterion E_1 to be an extension of the K-Means criterion E (1) on data presented as an arbitrary matrix of pairwise similarity coefficients.

For matrices with small number of rows and small desired number of metaclasses, the optimal partition may be computed by a brute force algorithm that examines all possible partitions. However, this rapidly becomes implausible as the number of rows and number of metaclasses increases.

Therefore, we suggest a heuristic method, P1, for computing a good partition. This method starts with an initial partition and cycles through all points $i = 1, \ldots, K$. In each cycle, for each point $i \in S_u$ it virtually moves i to each S_v, $v = 1, \ldots, m$,$v \neq u$. It finds S_v such that moving that would lead to greatest increase in value of E_1. If a move that increases E_1 cannot be performed, i remains in S_u. If no point is moved in a cycle, the algorithm terminated.

Description of P1: It is necessary to put some effort into implementation of P1 in order for it to to be efficient. Let us make the following definitions:

$$\psi(i, S_u) = b_{ii} + \sum_{j \in S_u, j \neq i} b_{ij},\ u = \overline{1, m},\ i = \overline{1, K}. \tag{15}$$

$$\delta(S_u) = \frac{1}{|S_u|} \sum_{i \in S_u} \psi(i, S_u)\ u\overline{1, m} \tag{16}$$

Using these definition we can re-express E_1 as:

$$E_1 = \sum_{u=1}^{k} \delta(S_u). \tag{17}$$

We note the following properties of $\psi(i, S_u)$:

$$\psi(i, S_u - j) = \psi(i, S_u) - b_{ij},\ \forall i \neq j. \tag{18}$$

$$\psi(i, S_u + j) = \psi(i, S_u) + b_{ij},\ \forall i \neq j. \tag{19}$$

Our algorithm works by computing a $K \times m$ matrix of partial sums of the initial partition:

$$C = ||c_{ij}|| = \psi(i, S_j) \tag{20}$$

The u and v columns of C are easily updated every time a point is moved via equations 18 and 19. This matrix, together with B, is used to evaluate utility of moving points to avoid computing E_1 for each case.

Let $\Delta(j, S_u, S_v)$ be the change in value of E_1 resulting from moving j from S_u to S_v. It is easy to see from (17) that:

$$\Delta E_1(j, S_u, S_v) = \delta(S_u - j) + \delta(S_v + j) - \delta(S_u) - \delta(S_v). \tag{21}$$

Proposition: 1 $\Delta(j, S_u, S_v)$ can be computed from C and B in $O(K)$ time.

The four terms on the right side of (21) can be easily computed from matrices C and B. The negative terms are computed on the basis of equation (15) in $O(K)$. As for the positive terms:

$$\begin{aligned}
\delta(S_u - j) = &\\
&\tfrac{1}{|S_u|-1} \sum_{i \in S_u, i \neq j} \psi(i, S_u - j) \\
&= \tfrac{1}{|S_u|-1} \sum_{i \in S_u, i \neq j} (\psi(i, S_u) - b_{ij}) \\
&= \tfrac{1}{|S_u|-1} \Big(\sum_{i \in S_u, i \neq j} \psi(i, S_u) - \sum_{i \in S_u, i \neq j} b_{ij} \Big) \\
&= \tfrac{1}{|S_u|-1} \Big(\sum_{i \in S_u, i \neq j} \psi(i, S_u) - (\psi(j, S_u) - b_{jj}) \Big) \\
&= \tfrac{1}{|S_u|-1} \Big(\sum_{i \in S_u} \psi(i, S_u) - \psi(j, S_u) - (\psi(j, S_u) - b_{jj}) \Big) \\
&= \tfrac{1}{|S_u|-1} \Big(\sum_{i \in S_u} \psi(i, S_u) - 2\psi(j, S_u) + b_{jj} \Big).
\end{aligned}$$

Similarly:

$$\delta(S_v + j) = \frac{1}{|S_v| + 1} \sum_{i \in S_v + j} \psi(i, S_v + j)$$

$$= \frac{1}{|S_v| + 1} \Big(\sum_{i \in S_v} \psi(i, S_v) + 2\psi(j, S_v) - b_{jj} \Big)$$

Thus both $\delta(S_u - j)$ and $\delta(S_v + j)$ can be computed via entries of matrices C and B in $O(K)$. ■

Theorem 1: Computational complexity of P1 is $O(cmK^2)$, where c is a finite positive integer.

Proof: Matrix C can be precomputed in $O(K^2)$, since all partial sums for a point together require $O(K)$.

Each of the four terms in (21) can be computed from matrix C in $O(K)$ summations (Proposition 1). In order to find the best move for point i we examine m clusters. Updating matrix C after a move requires updating entries in two columns: $O(K)$. Therefore, the complexity of a single cycle is $K(O(mK) + O(K)) = O(mK^2)$.

During each cycle, the value of E_1 increases, i.e $\Delta(j, S_u, S_v) > 0$. Since the matrix B is finite, there can be only be finitely many moves resulting in a positive $\Delta(j, S_u, S_v)$. Therefore, if ϵ is the infimum of $\Delta(j, S_u, S_v) > 0$ over all possible S_u,S_v and j, it follows that $\epsilon > 0$. Since

$$0 < E_1 \leq \sum_{i,j=1,\ldots,k} b_{ij}. \tag{22}$$

and each cycle of P1 increases E_1 by at least ϵ, there can be no more than $c = \frac{1}{\epsilon} \sum\limits_{i,j=1,\ldots,K} b_{ij}$ cycles. So P1 is guaranteed to terminate in finite amount of time: $O(cmK^2)$. ■

The pseudocode for P1 is given in Algorithm 7.

Algorithm 7 Partitioning Confusion Matrix: Method P1

Require: A symmetric $K \times K$ matrix B with non-negative values, m - desired number of metaclasses

Obtain an initial partition of B into m clusters $S_1, \ldots, S_m$.
Compute $K \times m$ matrix $C = ||c_{ij}|| = ||\psi(i, S_j)||$.
while At least one object changed clusters **do**
    for $i = 1, \ldots, K$ **do**
        if $|S_u| > 1$, where $i \in S_u$ **then**
            Find cluster j such that: $j = \text{argmax}_{v=1,\ldots,m} \Delta E_1(i, S_u, S_v)$
            if $i \neq j$ **then**
                Move i to S_j
                Update c_{xu} and c_{xj}, $\forall x = 1, \ldots, K$,$x \neq i$.
            end if
        end if
    end for
end while

We suggest another approach to partitioning confusion matrix. This approach is an even closer analog of K-Means than P1. We find the center s_u of cluster S_u via the formula:

$$s_u = \text{argmax}_{i \in S_u} \sum_{j \in S_u} b_{ij}. \tag{23}$$

The optimization criterion to be maximized can then be described as:

$$E_2 = \sum_{u=1}^{k} \max_{i \in S_u} \sum_{j \in S_u} b_{ij}. \tag{24}$$

An algorithm P2 for optimizing E_2 works like an iterative K-means algorithm. We find centers of each cluster. Then for each point (that is not the only point in its cluster), we find the cluster with the closest center. If the point does not belong to that cluster, we move it, and update the centers. The algorithm terminates when no point can be moved. A precise method for optimizing (24) is described in Algorithm 8.

Algorithm 8 Partitioning Confusion Matrix: Method P2

Require: A symmetric $K \times K$ matrix B with non-negative values, m - desired number of clusters

Obtain an initial partition of B into m clusters
for $u = 1, \ldots, m$ **do**
  compute s_u - center of S_u
end for
while At least one object changed clusters **do**
  for $i = 1, \ldots, K$ **do**
    if $|S_u| > 1$, where $i \in S_u$ **then**
      Find cluster $j = \text{argmax}_{v=1,\ldots,m}\, b_{is_v}$
      if $i \neq j$ **then**
        Move i to S_j
        Recompute centers of S_u and S_j
      end if
    end if
  end for
end while

The present section described methods of analyzing confusion matrix in order to obtain meta-classes. Once the partition into meta-classes is available, we can construct a classifier as described earlier in section 5.2.

5.3.2 Constructing a Hierarchical Classifier Based on Meta-Classes

We also suggest an alternative method for constructing hierarchical classifiers based on confusion matrix analysis. We can partitioning each class into k clusters, training first-level classifier on $k \cdot K$ clusters and using cross-validation to obtain a cluster vs. cluster confusion matrix. Ideally, this approach will cause each class to split into different regions based on their similarity to a clusters belonging to other classes. The resulting clusters will be easier to group than original classes. We then train classifiers to separate clusters that fall into the same group. We call this approach ReUsing First Level (RUFL), to distinguish it from the

RTFL approach of Section 5.2. Note that when $k = 1$, the classifier structure is the same as that of classifier described in Section 5.2, in Algorithm 5.

A specific variant, V, of our method deserves special attention because it can be performed with much less computational effort than the general case. This happens when $m = K * k$. In this case each class is partitioned into k clusters and a R_1 classifier is trained to distinguish them. However, after that, the label assigned by R_1 uniquely specifies the class of a point. Therefore we do not need to compute and analyze a confusion table, or construct the second level classifier. Additionally, when $k = 1$, this variant becomes equivalent to a global classifier.

Training and classification pseudocode for RUFL approach are given in Algorithms 9 and 10. The V variant is described in Algorithm 11 and 12.

Algorithm 9 Classifier Training with Confusion Matrix Analysis: RUFL Method

Require: A set W, number of clusters $k > 0$, number of metalasses $m > 0$.

for $j = 1, \ldots, K$ **do**
    Partition L_j into k clusters.
end for
Use cross-validation to obtain cluster confusion matrix A.
Train R_1 using all training data to recognize $K \cdot k$ clusters
Partition clusters based on A in m groups, to minimize the number of mistakes outside of groups
for $i = 1, \ldots, m$ **do**
    Train R_{2i} to separate clusters that fall into group m
end for

Algorithm 10 Classification with Confusion Matrix Analysis: RUFL Method

Require: A point x.

Let $i = R_1(x)$, $i = \overline{1, k \cdot K}$.
Let j be the metacluster to which cluster i belongs.
Let $u = R_{2j}(x)$, $u = \overline{1, K}$.
Return u.

Algorithm 11 Classifier Training with Confusion Matrix Analysis: V Method

Require: A set W, an integer $k > 0$.

for $j = 1, \ldots, K$ **do**
    Partition L_j into k clusters.
end for
Train R_1 using all training data to recognize $K \cdot k$ clusters

As described in Section 5.2, Algorithm 5 RTFL has previously been used only with $k = 1$. We also generalize RTFL approach by considering other values of k.

Algorithm 12 Classification with Confusion Matrix Analysis: V Method

Require: A point x.

1: Let $i = R_1(x)$, $i = \overline{1, k \cdot K}$.

2: Return class of cluster i.

k	$k=1$		$k=2$		$k=3$		$k=4$	
	P1	P2	P1	P2	P1	P2	P1	P2
$m=2$	90.81	90.81	89.81	88.81	90.76	90.14	91.95	89.67
$m=3$	90.05	90.38	89.52	88.67	88.38	89.29	92.05	89.43
$m=5$	90.76	89.90	88.05	89.19	88.24	88.29	91.33	89.57
$m=K*k$	90.95	90.95	87.76	87.76	87.14	87.14	88.33	88.33

Table 6: Image Dataset, Confusion Matrix Analysis, RUFL

5.4 Experiments

We conduct experiments for number of clusters per class $k = 1, 2, 3, 4$ and for the number of metaclasses $m = 2, 3, 5$. In order to obtain confusion matrix we use 5-fold cross-validation on the training set. The whole training set is then used to construct the first level classifier R_1. For each combination of parameters k and m we use both P1 and P2 partitioning methods, starting with the same initial conditions.

The V variant ($m = k \times K$) was implemented separately, since it does not require analysis of confusion matrix or construction of second-level classifiers.

As in all previous experiments, we use multi-class linear SVM mode of BSVM [2] (parameter settings "-s 2 -t 0").

5.5 Discussion of Results

Results of using RUFL approach are given in Tables 6-9. For each value of k and m we compare results when using partitioning methods P1 and P2. The last row of each table contains the result of using V method (when $m = K * k$). For $k = 1$, the method is identical to global classifier. Results in all other cells of the Table can be compared against it.

Performance on Image Dataset is described in Table 6. The accuracy varies from several

k	$k=1$		$k=2$		$k=3$		$k=4$	
	P1	P2	P1	P2	P1	P2	P1	P2
$m=2$	91.02	91.68	91.17	94.45	92.74	94.97	90.82	94.23
$m=3$	91.11	91.65	93.60	95.43	92.57	95.80	91.51	93.34
$m=5$	92.65	91.22	95.31	95.31	94.00	95.65	91.37	95.23
$m=K*k$	90.85	90.85	95.17	95.17	95.20	95.20	95.03	95.03

Table 7: Pendigits Dataset, Confusion Matrix Analysis, RUFL

k	$k=1$		$k=2$		$k=3$		$k=4$	
	P1	P2	P1	P2	P1	P2	P1	P2
$m=2$	83.00	82.90	82.60	82.15	82.70	84.55	82.50	84.25
$m=3$	82.95	83.15	84.20	82.45	83.00	84.40	83.00	84.80
$m=5$	82.55	82.65	83.40	82.10	83.10	84.90	82.80	84.55
$m=K*k$	82.6	82.6	82.65	82.65	83.65	83.65	83.95	83.95

Table 8: Satimage Dataset, Confusion Matrix Analysis, RUFL

k	$k=1$		$k=2$		$k=3$		$k=4$	
	P1	P2	P1	P2	P1	P2	P1	P2
$m=2$	41.56	41.77	41.56	39.83	38.96	38.96	39.18	39.18
$m=3$	40.26	40.26	43.29	39.18	41.77	37.01	38.74	40.04
$m=5$	38.74	40.91	42.64	39.39	42.21	40.04	35.06	37.88
$m=K*k$	37.45	37.45	41.99	41.99	43.94	43.94	39.18	39.18

Table 9: Vowel Dataset, Confusion Matrix Analysis, RUFL

k	$k=1$		$k=2$		$k=3$		$k=4$	
	P1	P2	P1	P2	P1	P2	P1	P2
$m=2$	83.48	89.19	76.90	82.71	77.00	89.43	75.86	74.33
$m=3$	90.10	81.10	85.14	85.57	64.71	85.81	77.95	78.10
$m=5$	88.24	88.71	83.76	87.95	78.52	86.05	87.24	84.90
$m=K*k$	90.95	90.95	87.76	87.76	87.14	87.14	88.33	88.33

Table 10: Image Dataset, Confusion Matrix Analysis, RTFL

k	$k=1$		$k=2$		$k=3$		$k=4$	
	P1	P2	P1	P2	P1	P2	P1	P2
$m=2$	65.07	91.60	63.12	84.39	75.76	91.74	72.24	84.71
$m=3$	71.44	90.34	72.47	83.96	76.39	90.94	73.21	88.85
$m=5$	80.13	91.42	72.38	93.51	77.27	89.42	64.52	90.42
$m=K*k$	90.85	90.85	95.17	95.17	95.20	95.20	95.03	95.03

Table 11: Pendigit Dataset, Confusion Matrix Analysis, RTFL

k	$k=1$		$k=2$		$k=3$		$k=4$	
	P1	P2	P1	P2	P1	P2	P1	P2
$m=2$	74.80	81.75	80.85	81.95	80.65	80.00	67.55	82.65
$m=3$	82.95	83.45	74.85	82.30	75.45	82.55	75.10	84.20
$m=5$	81.05	82.60	80.05	82.20	75.60	84.95	75.50	84.30
$m=K*k$	82.6	82.6	82.65	82.65	83.65	83.65	83.95	83.95

Table 12: Satimage Dataset, Confusion Matrix Analysis, RTFL

k	$k=1$		$k=2$		$k=3$		$k=4$	
	P1	P2	P1	P2	P1	P2	P1	P2
$m=2$	38.96	36.36	32.03	30.74	26.84	26.84	27.49	27.49
$m=3$	39.39	39.39	31.60	35.28	30.09	27.27	26.19	36.58
$m=5$	34.63	41.99	36.15	33.55	30.74	33.33	27.49	32.03
$m=K*k$	37.45	37.45	41.99	41.99	43.94	43.94	39.18	39.18

Table 13: Vowel Dataset, Confusion Matrix Analysis, RTFL

percent below (88.05 - $(k, m, P) = (2, 5, P1)$) that of a global classifier, to comparable or slightly better (92.05 - $(k, m, P) = (4, 3, P1)$). P1 is better than P2 in 8 cases out of 12, and worse in 3. The V method gives performance that is worse than that of a global classifier.

Results on Pendigit Dataset are in Table 7. The V variant performs well but does not have the best results in for any k. It does perform better than P1 for $k = 3, 4$. Best value for P1 is 95.31 ($k = 2$,$m = 5$), while for P2 it is 95.65 ($k = 3$,$m = 5$). Here P2 is better than P1 in 11 out of 12 cases.

Table 8 contains results of RUFL method on Satimage Dataset. Majority of results are better than those of a global classifier. The V method performs well again, but worse than the best results for each k. Best result for P1 is 84.2 for $k = 2$,$m = 2$, while for P2 it is 84.9 with $k = 3$ and $m = 5$. Overall, P1 outperforms P1 only in 4 cases.

Vowel Dataset results are presented in Table 9. Almost all results are better than those of a global classifier. The V variant performs better than the global classifier. It outperforms P1 for $k = 2$ and P2 for $k = 2, 3$. It also has the best overall result, 43.94, with $k = 3$. Best result for P1 is 43.29 ($k = 2$,$m = 2$). Best result for P2 is 41.77 ($k = 3$,$m = 2$ and $k = 1, m = 1$). P1 does better than P2 in 5 cases, and as well as P2 in 3 cases.

We will not discuss results of the RTFL approach, given in Tables 10-13, in great detail, because in almost all cases the accuracy of this method was lower than of RUFL method with the same parameters, and, in many cases, lower than of the global classifier.

5.6 Conclusions and Future Work

Overall, it appears that RUFL method can improve on the global classifier and the somewhat simpler variants of itself, though in some cases RUFL's results are worse than of a global classifier. The comparison with RTFL is decidedly in RUFL's favor: in addition to being less expensive than RTFL, it almost always outperforms it. This confirms our intuition that it is necessary for first level classifier to make mistakes distirbuted like in confusion matrix, and therefore that the same classifier as was used to obtain confusion matrix should be used for classification.

It is difficult to predict when one of the partitioning methods will be better than another. However on average P2 produces better results. Overall, P1 outperformed P2 in 14 cases, and did as well in 4 out of 36 total. This suggests that P2 is a better method. The effect of quality of the partition (in analysis of confusion matrix) on the classifier accuracy should be a subject of further study.

The simpler variant of our approach, V method, shows very good performance, while requiring less computational effort than RUFL. However it does not usually produce the best results on the dataset.

Some methods for choosing values of k and m (based for example on number of classes and training points) need to be suggested and examined.

Our approach can potentially be combined with Hierarchy with Global Classifier idea (Section 4.3) by using either global classifier R_0 trained separately, or by using R_1 classifier to determine a cluster, and thereby a class, of a point if local rule does not appear reliable.

6 Constructing Cluster Similarity Features

6.1 Introduction

One of the most important aspects of machine learning is the choice of feature space. A classification problem will be easy in a good space, while in a bad space even the most noticeable patterns may become obscured. The time and space efficiency of learning algorithms are also often strongly dependant on the size and quality of the feature space.

Our experiments, described in Sections 4 and 5 suggest that dividing our data into smaller groups, and thereby partitioning a problem into smaller problems, leads to improved performance, with very little cost added for the higher-level problem. Therefore, it seems reasonable that constructing features indicative of the points similarity to the clusters would also improve performance. We suggest a method for constructing features that is related to "concept indexing" method.

Recently, a "concept indexing" method was successfully applied to text clustering, text classification and information retrieval ([4], [10],[13]). This method involves clustering data or, in a supervised variant, using the external partition induced by class labels, followed by computation of centroids of the clusters and projection of all data points onto these centers.

In this section we present a modification of the concept indexing approach. The two main differences are that, firstly, instead of projecting onto centers we compute a non-linear function of similarity to centers, and, secondly, that we combine the representation in terms of similarities with the original features. As our results demonstrate, this approach performs better than using only the similarity space.

All of our experiments are conducted on the four datasets (Image, Pendigit, Satimage and Vowel) described in Section 3.

This section is structured as follows. We first discuss existing work. We then provide detailed description of our method, experiments and their results. The last subsection contain our conclusions.

6.2 Existing Work

6.2.1 Unsupervised Concept Indexing

Concept indexing ([10],[13]) relies on clustering algorithms to produce a partition of training data, centroids of which are then computed and used as axes for a new space. As described in the [10], concept indexing consists of the following steps:

- Step 0: Represent all data points as unit length vectors.
- Step 1: Partition dataset W into k clusters, using K-Means procedure. This produces clusters S_i, $i = 1, \dots, k$, with centers s_i.
- Step 2: Compute a new k-dimensional representation of the data in space Y (from original space X) using the map:

  $$M : X \mapsto Y,\ M(x) =< x^T \cdot s_1, \dots, x^T \cdot s_k > . \tag{25}$$

  Note that the space Y can have smaller dimensionality than X.
- Step 3: Train and test a classifier on data projected into a new space Y.

6.2.2 Supervised Concept Indexing

Supervised concept indexing [10] differs from unsupervised one only in Step 1, where, instead of partitioning the whole set W, each class L_m, $m = 1, \dots, K$ is partitioned separately, so that all resulting clusters consist only of points of one class. In other words, a partition of W into clusters S_i,$i = 1, \dots, k \cdot K$ has to satisfy:

$$L_m = \bigcup_{j=1,\dots,k} S_{(m-1)\cdot k+j},\ m = 1, \dots, K, \tag{26}$$

6.3 Our Approach

As mentioned in Section 6.1, instead of projecting data onto cluster centers, we compute a non-linear function of similarity between a point and a center. The particular similarity function used in given in the formula 27:

$$\rho(x, y) = e^{-\frac{||x-y||^2}{\alpha}}. \tag{27}$$

In the above formula, α is a free parameter that can be used to specify the range of significant similarities. If α is small then the similarities between most points and centers will be small, while for a few very close points they will be large. Increasing α would lead to more similarities being significant. For our experiments, we chose α to be the average

k	X	SV	Y_s	SV_s	Y_{os}	SV_{os}
10 clusters	90.95	68	79.29	128	90.24	69
50 clusters	90.95	68	87.14	76	88.95	66
supervised 1 (7 total)	90.95	68	84.05	100	90.33	67
supervised 5 (35 total)	90.95	68	87.38	82	89.81	66

Table 14: Image Dataset, Concept Indexing Results

k	X	SV	Y_s	SV_s	Y_{os}	SV_{os}
10 clusters	90.85	1016	88.79	1501	95.54	559
50 clusters	90.85	1016	97.14	452	97.08	380
supervised 1 (10 total)	90.85	1016	88.56	1492	95.31	593
supervised 5 (50 total)	90.85	1016	96.86	450	96.94	367

Table 15: Pendigit Dataset, Concept Indexing Results

distance from a point to a center. This way each center has some influence over most points. The exact formula for computing α is given below:

$$\alpha = \frac{1}{|W| \cdot k} \sum_{x \in W} \sum_{i=1,\ldots,k} ||x - s_i||^2. \tag{28}$$

Therefore, in Step 2 of concept indexing, we map data into space Y_s as follows:

$$M_s : X \mapsto Y_s,\ M_s(x) =< \rho(x, s_1), \ldots, \rho(x, s_k) > . \tag{29}$$

In order to improve the accuracy of classification, we suggest combining original space with the similarity space Y_s. In other words we map data into a space $Y_{os} =< X, Y_s >$, using mapping:

$$M_{os} : X \mapsto Y_{os},\ M_{os}(x) =< x, M_s(x) > \tag{30}$$

that takes x from a d-dimensional space X into a $(d + k)$-dimensional space Y_{os}.

6.4 Experiments

The results of experiments with concept indexing are presented in Tables 14-17. These Tables are organized as follows. The first row contains headings. The next two rows contain results

k	X	SV	Y_s	SV_s	Y_{os}	SV_{os}
10 clusters	82.6	1409	83.50	1473	87.20	1217
50 clusters	82.6	1409	83.65	1545	88.10	1160
supervised 1 (6 total)	82.6	1409	82.30	1584	87.10	1222
supervised 5 (30 total)	82.6	1409	85.65	1385	87.70	1175

Table 16: Satimage Dataset, Concept Indexing Results

k	X	SV	Y_s	SV_s	Y_{os}	SV_{os}
10 clusters	37.45	456	52.81	410	55.41	297
50 clusters	37.45	456	56.06	225	57.36	209
supervised 1 (11 total)	37.45	456	55.19	412	51.30	309
supervised 5 (55 total)	37.45	456	58.01	216	58.01	207

Table 17: Vowel Dataset, Concept Indexing Results

of unsupervised concept indexing, while the last two rows contain results of supervised concept indexing. The first column contains the value of parameter k. The remaining 6 columns form three pairs. Each pair of columns provides the accuracy (the percentage of test points classified correctly) and the number of support vectors used in a particular space. The first pair contains results in the original space X (therefore the values in these columns do not change with k). The next pair relates to similarity space Y_s only, while the third pair shows results in combined space Y_{os}.

The data in original space was normalized. The data in Y_s and Y_{os} was also normalized before training and testing SVM. This was done to scale all variables to the same magnitude. This is especially important for Y_{os}, where original features are combined with inter-point similarities.

Training set of each dataset is used for estimating mean and variance of each variable, for constructing the mapping into similarity space and combined space. The same training set is then projected into the resulting space, and classifier is trained on this set. Separate test sets are used to evaluate the classifier's performance on each dataset in each of the spaces.

We experiment both with unsupervised and supervised concept indexing schemes. In the unsupervised case, we take the whole training set and create a partition into k clusters, using K-Means algorithm with random initial partition. Values of k used were 10 and 50. In the supervised case, we conduct a single experiment for each of $k = 1$ and $k = 5$. The resulting number of clusters is indicated in the first column in parenthesis.

As mentioned in Section 2.3, we used multi-class linear SVM mode of BSVM [2].

6.5 Discussion of Results

The results can be split into two groups: those on Image dataset (Table 14) and the rest. The reason for this is that in Table 14 all the concept indexing results are worse than results in the original space, while for other datasets this is not the case.

Let us discuss Image dataset results first. Here performance in the original space is quite good (90.85) and therefore difficult to beat. Considering that the training set only has 210 points, we have to conclude that the original representation is a good one. Results using similarity space alone are significantly worse (79.29 - 87.38), while those using combined space are somewhat worse (88.95 - 90.33) than original space. This happens despite the fact that the number of support vectors in combined space is smaller than in the original space in all cases except unsupervised with $k = 10$. Increase in the number of clusters leads to decrease in classification accuracy when using Y_{os}. These signs suggest that there may not be enough training data for concept indexing to construct good new features: when $k = 50$, the average number of points in a cluster is only 4.2, implying that at least some of the clusters do not correspond to dense regions in data distribution.

The results on other datasets suggest that our method can significantly improve accuracy. We note that in all of these datasets the size of training set is larger than in Image dataset (528 in Vowel and several thousand for Pendigit and Satimage).

Results for Pendigits dataset are in Table 15. The results in combined space Y_{os} are significantly better than those in the original space. The accuracy improves from 90.85 to [95.54, 97.08]. The results in similarity space Y_s vary from slightly worse (88.56 and 88.79) to significantly better (96.86 and 97.14), as the number of clusters increases. We can see that the best result in Y_s is in fact better than the best in Y_{os}, but results in Y_{os} are much more consistent. For both Y_s and Y_{os} the best result is achieved with unsupervised approach, $k = 50$. However, the results of supervised approach are very close to those of unsupervised.

Results for Satimage dataset (Table 16) are qualitatively similar to those for Pendigits. Accuracies in similarity space Y_s vary from slightly worse to better (82.3 - 85.65) better than in the original (82.6), while in combination space they are significantly better (87.1 - 88.1). Here results in Y_{os} are always better than in Y_s. The amount of improvement in Y_{os} is approximately the same as in Pendigit dataset. The best result in Y_{os} is achieved with unsupervised method and $k = 50$. Results with supervised clustering are comparable to those with unsupervised clustering in Y_{os} (taking into account the difference in the total number of clusters used). For Y_s however, the best result is with supervised approach and $k = 5$.

Table 17 contains results of experiments on the Vowel dataset. Here the results in the original space are rather poor (37.45). Use of similarity space leads to significantly higher results (52.81 - 58.01) than in the original space. Combining similarity space with the original space leads to higher results than similarity space alone with unsupervised clustering (55.41 and 57.36) but not with supervised clustering. The best results in both Y_s and Y_{os} are equal to 58.01 and are achieved with supervised approach for $k = 5$.

We note that in all Tables, the number of support vectors in combined space Y_{os} was less than that used in original space X and in similarity space Y_s alone, while that number in similarity space Y_s was sometimes less and sometimes greater than in the original space. The number of support vectors can be seen as an indicator of the accuracy of classier on the training set [14] (few support vectors imply better accuracy). It is also important for efficiency reasons, since the number of support vectors is directly proportional to classification time. Therefore these observations support the claim that Y_{os} is usually the better space for classification.

6.6 Conclusions and Future Work

Our results demonstrate that representing data in a combination Y_{os} of similarity space Y_s and original space X almost always leads to a better classification accuracy than using either one of these representations by itself. Also, the number of support vectors used in combined space is less than that in the original or similarity space.

Increasing dimensionality of representation (creating more clusters) usually leads to better classification results. This is likely due to the nature of SVM, that can exploit increase in dimensionality to construct better classifiers, while avoiding overfitting the data.

We also note that overall the use of supervised concept indexing is comparable with unsupervised one. It is unclear whether one of these methods is better than the other.

Our results allow us to conclude that in most cases our approach to concept indexing can be of benefit. However, our method may fail are when the accuracy in the original space is already high and there is little training data available. We surmise that in such situations cluster centers do not correspond to meaningful dense region in data and the resulting features do not carry additional useful information but are noisy.

If sufficient amount of training data is available, it seems worth the effort to experiment with classifiers both in original space and in combined space Y_{os}. When the original space X has high dimensionality as happens in text mining or bioinformatics application, using similarity space Y_s alone may be considered, since both literature ([10],[13]) and our results suggest that use of Y_s by itself can also lead to improved accuracy compared to original space.

These observations allow us to suggest an "algorithm" for using concept indexing. Given a new training set W, evaluate performance of a classifier in the original space. If dimensionality of data is high, further experiments should focus on Y_s, otherwise Y_{os} should be explored. If the accuracy in the original space X is low, experiments with various values of k should be done. If the accuracy in X is high, then experiments with Y_s or Y_{os} should be considered only if sufficient training data is available.

In this work we did not experiment with other similarity functions that can be used for constructing new features. We also did not consider the question of whether features based on distances to the clusters would proves as effective as similarity-based measures. These questions can only be answered by further empirical studies.

7 Comparative Analysis of the Three Approaches

In this work we examined two approaches to using cluster analysis in constructing hierarchical classifiers. We suggested improvements and new methods for each of these approaches. We also devised a method for creating new features based on cluster analysis of data.

These three distinct approaches were empirically evaluated on the four datasets from UCI. Our results and analysis show that these methods are applicable in different situation and appear to vary greatly in their computational cost and level of improvement that they provide. In this section we attempt to give recommendations for their use.

The approach that appears to give greatest improvement in accuracy is combining original feature space with similarity space. This approach is also least expensive computationally. This allows the researcher to easily experiment with several possible values for parameter k and with original space to decide whether combination should be used or whether the original space provides sufficient accuracy. Furthermore, a good value of k can be chosen with cross-validation or a hold-out set.

Once a mapping is constructed, any new data can be easily projected to a new space and classified. The final classifier is a global multi-class SVM. This approach can furthermore be combined with hierarchical approaches or confusion matrix approach.

A significant limitation of this approach is that it increases the dimensionality of data. For data that is already high-dimensional this might make training a classifier and labeling new points rather expensive. In this case we would suggest using a (relatively) low-dimensional similarity space alone. Since parameter k in this case indicates the resulting dimensionality, this approach would allow researcher to choose a space with desirable accuracy. Experiments on text data ([10],[13]) show that for sparse high-dimensional data similar techniques are effective.

If constructing a new representation does not improve accuracy, or does not do it to a sufficient degree, using one of hierarchical methods of Section 4 (SH or HGC) may improve the situation. The downside of these methods is the increased number of classifiers that are needed.

An alternative method of constructing hierarchy is via analysis of confusion matrix. However our results seem to indicate that the benefits of this approach (improvements in accuracy) are less than of the other hierarchical methods discussed here, while the computational cost is at least as high (where hold-out set is available to estimate confusion matrix), or much higher (when cross-validation is done for that purpose).

The general effectiveness of combining our method of feature extraction with our hierarchical methods still needs to be explored.

The results presented here are preliminary. Until further research confirms or rejects our conclusions, these recommendations should be taken with a grain of salt. What is clear however, is that there is a great potential for combining clustering methods with supervised learning.

8 Acknowledgments

The authors would like to thank Dr. Kulikowski for helpful discussions and comments.

References

[1] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[2] 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`.

[3] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learning Theory*, pages 35–46, 2000.

[4] I. Dhillon and D. Modha. Concept decomposition for large sparse text data using clustering. Technical report, IBM Almaden Research Cneter, 1999.

[5] J.-X. Dong, A. Krzyzak, and C. Y. Suen. Local learning framework for recognition of lowercase handwritten characters. In *Proceedings of MLDM 2001*, pages 226–238, 2001.

[6] S. Eschrich and L. O. Hall. Soft partitions lead to better learned ensembles. In *Proceedings of NAFIPS*, pages 406–411, 2002.

[7] W. D. Fisher. On grouping for maximum homogeneity. *J.Am.Stat.Assoc*, 53:789–798, 1958.

[8] S. Godbole. Exploiting confusion matrices for automatic generation of topic hierarchies and scaling up multi-way classifiers. Technical report, IIT Bombay, 2002.

[9] S. Godbole, S. Sarawagi, and S. Chakrabarti. Scaling multi-class support vector machines using inter-class confusion. In *Proceedings of SIGKDD'02*, 2002.

[10] G. Karypis and E.-H. Han. Fast supervised dimensionality reduction algorithm with applications to document categorization and retrieval. In A. Agah, J. Callan, and E. Rundensteiner, editors, *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, pages 12–19, McLean, US, 2000. ACM Press, New York, US.

[11] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:128–137, 1982.

[12] 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.

[13] M. Sasaki and K. Kita. Vector space information retrieval using concept projection. In *Proceeding of the 19th International Conference on Computer Processing of Oriental Languages*, pages 73–76, 2001.

[14] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 2nd edition, 1998.