Local Decomposition for Rare Class Analysis

Junjie Wu[1], Hui Xiong[2], Peng Wu[1], Jian Chen[1]

[1] Research Center for Contemporary Management, Key Research Institute of Humanities and Social Sciences at Universities, Tsinghua University, China, {wujj,wup2,chenj}@sem.tsinghua.edu.cn
[2] MSIS Department, Rutgers, the State University of New Jersey, USA, hxiong@andromeda.rutgers.edu

ABSTRACT

Given its importance, the problem of predicting rare classes in large-scale multi-labeled data sets has attracted great attentions in the literature. However, the rare-class problem remains a critical challenge, because there is no natural way developed for handling imbalanced class distributions. This paper thus fills this crucial void by developing a method for Classification using lOcal clusterinG (COG). Specifically, for a data set with an imbalanced class distribution, we perform clustering within each large class and produce sub-classes with relatively balanced sizes. Then, we apply traditional supervised learning algorithms, such as Support Vector Machines (SVMs), for classification. Indeed, our experimental results on various real-world data sets show that our method produces significantly higher prediction accuracies on rare classes than state-of-the-art methods. Furthermore, we show that COG can also improve the performance of traditional supervised learning algorithms on data sets with balanced class distributions.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier Design and Evaluation*

General Terms

Algorithms, Experimentation

Keywords

Rare Class Analysis, K-means Clustering, Support Vector Machines, Local Clustering

1. INTRODUCTION

Classification provides insight into the data by assigning objects to one of several predefined categories. An emerging critical challenge for classification is to address so-called "imbalanced classes" in the data. Specifically, people are interested in predicting rare classes in the data sets with imbalanced class distributions. For example, in the domain of network intrusion detection, the number of malicious network activities is usually very small compared to the number of normal network connections. It is crucial and challenging to build a learning model which has the prediction power to capture future network attacks with low false positive rates. Indeed, rare class analysis is often of great value and is demanded in many real-world applications, such as the detection of oil spills in satellite radar images [20], the prediction of financial distress in enterprises [33], and the diagnoses of rare medical conditions [25].

To meet the above challenge, considerable research efforts have been focused on the algorithm-level improvement of the existing classifiers for rare class analysis. Two promising research directions are the use of re-sampling techniques and cost-sensitive learning [29]. These two methods indeed show encouraging performances in some cases by directly or indirectly adjusting the class sizes to a relatively balanced level. Nevertheless, in this paper, we reveal that the class imbalance problem is strongly related to the presence of complex concepts (inherent complex structures) in the data. For imbalanced data sets with complex concepts, it is often not sufficient to simply manipulate the class sizes. In fact, our experimental results show that adjusting the class sizes alone usually can improve the predictive accuracy of the rare classes slightly, but at the cost of seriously decreasing the accuracy of the large classes. As a result, we need to develop a classification method which follows two criteria.

1. The ability to divide imbalanced classes into relatively balanced classes for classification.

2. The ability to decompose complex concepts within a class into simple concepts.

Indeed, this paper fills this crucial void by designing a method for classification using local clustering (COG). Specifically, for a data set with an imbalanced class distribution, we perform clustering within each large class and produce sub-classes with relatively balanced sizes. Then, we apply traditional supervised learning algorithms, such as SVMs, for classification. Since the clustering is conducted independently within each class but not across the entire data set, we call it ***local clustering***, which is the essential part of our

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'07, August 12–15, 2007, San Jose, California, USA.
Copyright 2007 ACM 978-1-59593-609-7/07/0008 ...$5.00.

COG method. By exploiting local clustering within large classes, we can decompose the complex concepts, e.g., non-linear-separable concepts for linear classifiers, into relatively simple ones, e.g., linearly separable concepts. Another effect of local clustering is to produce subclasses with relatively uniform sizes. In addition, for data sets with highly skewed class distributions, we further integrate the over-sampling technique into the COG scheme and propose the COG with over-sampling technique (COG-OS).

The merit of COG lies in three aspects. First, COG has the ability to divide imbalanced classes into relatively balanced and small sub-classes, and thus provide the opportunities in exploiting traditional classification algorithms for better predicting rare classes. Second, similar to the re-sampling schemes, COG is not a "bottom-level" algorithm but provides a general framework which can incorporate various existing classifiers. Finally, COG is especially effective on improving the performance of linear classifiers. This is noteworthy, since linear classifiers have shown their unique advantages, such as simplicity and understandability, higher executive efficiency, less parameters, less generalization errors [12, 28], in many cases.

We have conducted extensive experiments on a number of real-world data sets. Our experimental results show that, for data sets with imbalanced classes, COG and COG-OS show much better performances in predicting rare classes than two popular re-sampling schemes as well as two state-of-the-art rule induction classifiers without compromising the prediction accuracies of large classes. In addition, for data sets with balanced classes, we show that our COG method can also improve the performance of traditional linear classifiers, such as SVMs, by decomposing the non-linear-separable concepts into linearly separable ones.

COG (Classification using lOcal clusterinG)
Input: TR: a training data set.
TE: a test data set.
LCF: a linear classifier, such as SVMs.
K: a vector specifies the number of local clusters for each class.
R: a vector specifies the over-sampling times for each class. (for COG-OS only)
Output: CM: the model built on TR.
CR: the prediction results.
Procedure:
Phase I: local clustering
1. for class i=1 to C //C represents #classes
2. clusterLabel(i)=Clustering(TR(i), K(i));
3. TR(i)*=changeLabel(TR(i), clusterLabel(i));
4. end for
Phase II: over-sampling (for COG-OS only)
5. for class i=1 to C
6. TR(i)**=replicate(TR(i)*, R(i));
7. end for
8. TR**=merge$_{i=1,\cdots,C}$(TR(i)**)
Phase III: training
9. CM=train(TR**,LCF);
Phase IV: testing/predicting
10. clusterLabel=predict(TE, CM);
11. predictLabel=convertLabel(clusterLabel);
12. CR=compareLabel(predictLabel, trueLabel(TE)); (for testing only)

Figure 1: The COG Algorithm

2. ALGORITHM DESCRIPTION

In this section, we first describe our methods: COG (Classification using lOcal clusterinG) and COG-OS (COG with Over-Sampling). Then, we present the details about how to perform COG by a simple example.

2.1 COG and COG-OS

In a nutshell, COG provides a general framework which can incorporate various *linear* classifiers and improve their classification performance on data sets with *non-linear-separable concepts* as well as *imbalanced class distributions* . As to COG-OS, it is an extended version of COG, which integrates the over-sampling technique into the COG scheme for the purpose of better predicting rare classes in data sets with extremely imbalanced class distributions.

Figure 1 shows the pseudo-code of COG containing four phases. In Phase I, we employ K-means clustering on class i ($i = 1, 2, \cdots, C$) according to the user preset cluster number $K(i)$, and change the instance labels of class i with the subclass labels provided by the K-means clustering, thus form a multi-class data set with $\sum_{i=1}^{C} K(i)$ subclasses. Since we do clustering inside every class (if necessary) but not across the entire data set, we call it ***local clustering***. Phase II is dedicated to COG-OS. In this phase, we replicate $R(j)$ times the instances of class j ($j = 1, 2, \cdots, C$), to form a more balanced data set. Phase III is straightforward; that is, we build the model on the modified data set using a user specified linear classifier. Phase IV is simply for testing; however, each instance from the test set will be assigned with a label of a subclass which must be converted into the label of the corresponding parent-class.

There are some points needed to be further addressed. Typically, we do local clustering and over-sampling (if necessary) on different classes; that is, in Phase I and II, the cluster number $K(i)$ is larger than 1 for the relatively large class i, while the replication ratio $R(j)$ is larger than 1 for the small or rare class j. In practice, we first assign $K(i)$ with a small number, e.g., four, on the large class i ($i = 1, \cdots, C$), then use $R(j)$ on the small class j ($j = 1, \cdots, C$) to adjust the data set to a relatively balanced situation.

Also, we must emphasize that COG and COG-OS have more impact on the performances of linear classifiers. We know that linear classifiers have various merits such as simplicity and understandability, higher executive efficiency, less parameters, less generalization errors, and so on [12, 28]. In contrast, although non-linear classifiers such as SVMs with RBF kernel can find sophisticated boundaries, their parameters are typically hard to specify, and the use of non-linear kernels can easily lead to overfitting [12]. Therefore, in this paper, we use SVMs with a linear kernel. Furthermore, non-linear-separable concepts in the data is a long-standing challenge for the use of linear classifiers. To this end, COG and COG-OS can strengthen the use of linear classifiers by decomposing the non-linear-separable complex concepts into linear-separable ones. By contrast, as shown in our experimental results (Section 4), COG shows no consistent improvements on non-linear classifiers such as decision trees and rule based learning algorithms.

While we use k-means as the clustering scheme in COG, the choices of clustering algorithms in COG is not limited to k-means. Any other clustering algorithm which can produce clusters with relatively balanced sizes, such as the EM algorithm, can also be used in COG. Finally, COG is efficient in terms of the computational performance. First, if K-means

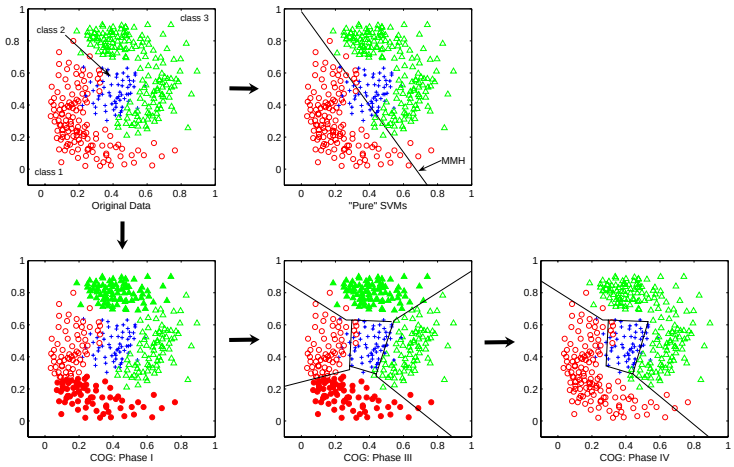

Figure 2: Illustration of the COG Procedure.

is used for the local clustering, the time required in the clustering phase is modest — basically linear in the number of data points [29]. Then in the training phase, as the number of classes increases, the time required is $\prod_{i=1}^{C} K(i) \times T$ where T is the time required for the training without local clustering. Since the number of relatively large classes in a data set is often very small, and in practice each $K(i)$ is usually assigned with a small number, e.g., four, we can expect to keep the computational cost of COG in the same level as the original classifier.

2.2 An Illustration of COG

Here, we use a synthetic data set with three classes and the SVMs classifier to illustrate the process of COG. The sizes of the three classes are 133, 60 and 165 respectively, and please note that the small class is non-linear-separable from the other two large classes, as shown in the "Original Data" subplot of Figure 2. The SVMs tool we used here is LIBSVM [5] with the linear kernel.

First, we build the classification model by simply applying SVMs on the original data set, and the results are shown in the pure SVM subplot. In this subplot, the solid line represents the maximal margin hyperplane (MMH) learned by SVMs algorithm. One interesting observation is that the instances of the small class, i.e., class 2, have totally "disappeared"; that is, they are all assigned to either class 1 or class 3 according to the only one MMH. This is due to the non-linear-separable concepts in the data.

Instead, we employ COG. First, we apply local clustering on class 1 and 3 respectively, given the cluster number is two. The clustering results can be seen in subplot "COG: Phase I". That is, class 1 and class 3 are divided into two subclasses respectively by K-means. Thus we obtain a modified data set with five relatively balanced and linear-separable classes. Next, we apply SVMs on this five-classes data set and get results shown in subplot "COG: Phase III". As can be seen, more MMHs appear in the model, which enables the model to identify the instances of class 2. Finally, for each instance, we convert its predictive label of some subclass into the label of the parent-class. This is equal to delete the MMHs separating the subclasses derived from a same parent-class, as indicated by the "COG: Phase IV" subplot. Therefore, by applying COG, we build up a more accurate model which can identify the instances from the small class among non-linear-separable concepts.

3. COG FOR RARE CLASS ANALYSIS

In this section, we illustrate why COG is especially effective on predicting rare classes using an example. First, we generate synthetic data sets for three different scenarios: data with simple concepts, data with non-linear-separable concepts, and data with complex concepts. In this example, we again use LIBSVM [5] with the linear kernel as the classifier in COG and COG-OS. Also, the synthetic data sets are two-class data sets with two dimensions, and the sizes of the rare and normal classes are 14 and 136, respectively.

3.1 Data with Simple Concepts

First of all, let's give an informal definition. We call that a concept is complex in data if the distributions of the instances from the two classes are too close to be separable by the linear classifiers. Therefore, in this scenario, we have two well-separated classes which represent a rather simple concept in the data, as shown in "Scenario I" of Figure 3. As can be seen, for this simple concept, pure SVM separates the rare and normal classes easily and precisely (the solid line represents the maximal margin hyperplane learned by the SVMs algorithm). This implies that the rare class problem will be inapparent in the case of simple concepts.

3.2 Data with Non-Linear-Separable Concepts

In Scenario II, we consider the case that data sets contain non-linear-separable concepts, which can seriously hinder the performance of linear classifiers. In the COG scheme, we exploit local clustering to divide non-linear-separable concepts into smaller linear-separable concepts. In this way, traditional linear classifiers can still work well in this scenario. In Figure 3, two subplots of "Scenario II" show the process of COG on handling non-linear-separable concepts. As can be seen in subplot II-I, pure SVM cannot effectively identify the rare class, since the instances of rare class and normal class are very close. After applying COG, however, we can divide the large class into two subclasses and form a data set with three linear-separable sub-classes, which can be easily learned by SVMs, as shown in subplot II-II.

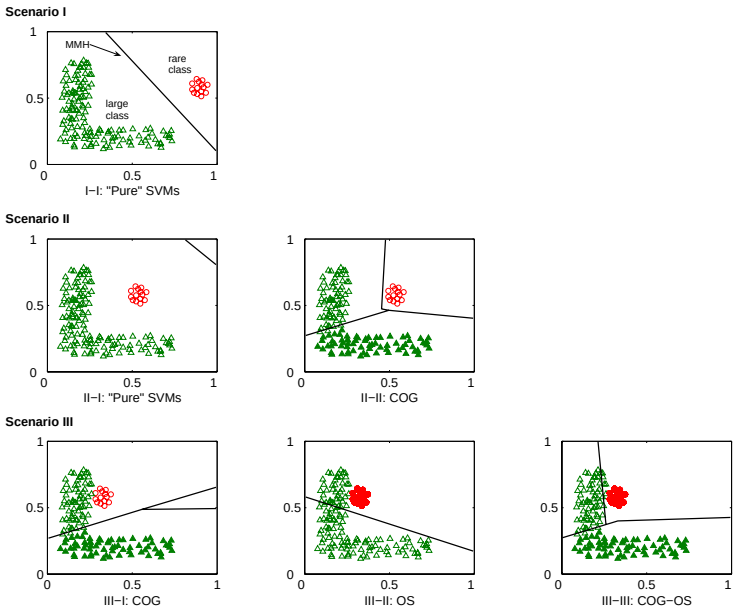

Figure 3: Illustration of Various Scenarios.

3.3 Data with Complex Concepts

In this subsection, we consider a more complicated case that data sets contain complex concepts. In other words, the instances of the rare class are adjacent to the instances of the normal class in the data, as shown by subplots of "Scenario III" in Figure 3.

In this scenario, COG alone seems not very helpful for rare class analysis. As can be seen in subplot III-I of Figure 3, COG cannot separate the rare class from one of the subclasses of the normal class. Also, in subplot III-II, we can see that the traditional over-sampling technique performs poorly for predicting rare class, since many instances of the normal class have been assigned to the rare class (the replicative time is set to be 8 on the rare class to obtain a relatively balanced distribution). However, COG with the over-sampling technique (COG-OS) can help SVMs successfully isolate the over-sampled instances of the rare class from the two sub-classes partitioned by K-means clustering on the normal class, as shown in the Subplot III-III of Figure 3.

Discussion. In summary, the rare class problem is strongly related to the presence of the complex concepts (inherent complex structures) in the data. The more complex the concepts are, the more significant the rare class problem is. By applying local clustering, COG can handle the rare class problem in the presence of the non-linear-separable concepts; by further incorporating the over-sampling scheme, COG-OS can handle the rare class problem in the case of more complex concepts.

4. EXPERIMENTAL EVALUATION

In this section, we present experimental results to validate the performance of the COG and COG-OS methods on balanced and imbalanced classification problems.

4.1 The Experimental Setup

Experimental Tools. We used four types of classifiers: support vector machines, Bayesian logistic regression, decision trees, and rule-based classifiers. Their corresponding implementations are LIBSVM [5], BMR [1], C4.5 [2], and RIPPER [7]. In all the experiments, default settings were used except that the kernel type of LIBSVM was set to be linear. Thus we have two linear classifiers, i.e., LIBSVM and BMR, and two non-linear classifiers, i.e., C4.5 and RIPPER.

Table 1: Some Notations.

US:	Under-sampling scheme.
OS:	Over-sampling scheme.
COG:	Classification using local clustering.
COG-OS:	COG with the over-sampling scheme.

Also, we applied K-means, a widely used clustering scheme which tends to produce clusters with relatively uniform sizes, as the clustering method in our COG method. During the K-means clustering, for data sets with relatively small number of dimensionality, squared Euclidean distance was used as the proximity measure; and for data sets with high dimensionality, however, the cosine similarity was used instead. This is due to the fact that Euclidean notion of proximity is not very meaningful for high-dimensional data sets, such as document data sets. Note that for each data set, K-means ran ten times and returned the best partitioning result.

Finally, some notations are given in Table 1. The default classifier used in these schemes is SVMs. If other classifiers are used instead, e.g, BMR in the COG scheme, we will explicitly denote it by "COG(BMR)".

Experimental Data Sets. For our experiments, we used a number of benchmark data sets that were obtained from different application domains. Some characteristics of these data sets are shown in Table 2. In the table, CV — Coefficient of Variation [9]— shows the dispersion of the class distribution for each data set. In general the larger the CV value is , the greater the variability is in the data.

UCI Data Sets. In our experiments, we used eight well-known benchmark data sets from UCI Repository [26]. Among them two data sets, `breast-w` and `pima-diabetes`, are binary data sets from the medical domain. The `breast-w` data set contains two types of results from real-world breast

Table 2: Some Characteristics of Experimental Data Sets.

Dataset	Source	#objects	#features	#classes	MinClassZize	MaxClassSize	CV
UCI Data Sets							
breast-w[§]	UCI	683	9	2	239	444	0.424
pima-diabetes	UCI	768	8	2	268	500	0.427
letter	UCI	20000	16	26	734	813	0.030
optdigits[†]	UCI	3823/1797	64	10	376/174	389/183	0.014/0.015
page-blocks	UCI	5473	10	5	28	4913	1.953
pendigits[†]	UCI	7494/3498	16	10	719/335	780/364	0.042/0.042
satimage[†]	UCI	4435/2000	36	6	415/211	1072/470	0.425/0.368
vowel[†]	UCI	528/462	10	11	48/42	48/42	0.000/0.000
Document Data Sets							
k1b	WebACE	2340	21839	6	60	1389	1.316
la12	TREC	6279	31472	6	521	1848	0.503
LIBSVM Data Sets							
fourclass	LIBSVM	862	2	2	307	555	0.407
german.numer	LIBSVM	1000	24	2	300	700	0.567
splice	LIBSVM	1000	60	2	483	517	0.048
SVMguide1	LIBSVM	3089	4	2	1089	2000	0.417

Notes: The numbers before and after "/" are for training and test sets respectively.
[§]: 16 instances with missing data have been deleted.
[†]: These data sets have been split into training/test sets by the sources.

cancer diagnosis, and the `pima-diabetes` data set is about the information of whether the patient shows signs of diabetes according to the WHO criteria. The rest six data sets are frequently used by the pattern recognition community. `letter`, `optdigits` and `pendigits` are data sets containing the information of handwritings; that is, `letter` has the letter information from A to Z, and the other two have the number information from 0 to 9. The `satimage` data set contains the multi-spectral values of pixels in 3×3 neighborhoods in a satellite image. The `page-blocks` data set contains the information of five types of blocks from a document page layout. And the last data set `vowel` was designed for the task of speaker independent recognition of the eleven steady state vowels of British English.

Document Data Sets. We also used high-dimensional document data sets in our experiments. The data set `k1b` was from the WebACE project [15]. Each document corresponds to a web page listed in the subject hierarchy of Yahoo!. The `la12` data set was obtained from articles of the Los Angeles Times that was used in TREC-5 [30]. The categories correspond to the *desk* of the paper that each article appeared and include documents from the entertainment, financial, foreign, metro, national, and sports desks. For these two document data sets, we used a stop-list to remove common words, and the words were stemmed using Porter's suffix-stripping algorithm [27].

LIBSVM Data Sets. Finally, we applied four binary data sets: `fourclass`, `german.numer`, `splice` and `SVMguide1` from the LIBSVM repository [5].

Please note that for any data set without an appointed test set, we did random, stratified sampling on it and had 70% samples as the training set and the rest as the test set.

4.2 The Effect of COG and COG-OS on Imbalanced Data Sets

In this subsection, we show how COG can improve the performance of linear classifiers on imbalanced data sets. As discussed in Section 3, the problem of imbalanced classes is related to the complex concepts in the data — such as non-linear-separable concepts for linear classifiers. The COG method is a natural solution to this problem: handling the non-linear-separable concepts as well as making the class sizes be relatively balanced. For data sets with highly imbalanced class sizes, such as binary data sets with rare classes, we apply COG with the over-sampling scheme (COG-OS). Specifically, for the large class of any binary data set, we did K-means clustering on it, and set the cluster number consistently to be 4; and for the rare class, we did over-sampling on it, and made the size be approximate to the average size of the partitioned large class. In this way, we can have much more balanced data sets. Also, the non-linear-separable concepts can be transformed into linearly separable concepts.

Table 3: Sampled Data Sets.

Data Set	Class	Sampling Ratio	#instances	CV
breast-w	1	0.20	48	1.14
	2	1.00	444	
pima-diabetes	1	0.20	54	1.14
	2	1.00	500	
fourclass	1	0.20	62	1.13
	2	1.00	555	
german.numer	1	0.20	60	1.19
	2	1.00	700	
splice	1	0.10	49	1.17
	2	1.00	517	
SVMguide1	1	0.05	53	1.34
	2	1.00	2000	

Also, we prepared the imbalanced data sets with rare classes via sampling on various binary data sets. Specifically, for each data set with two classes, we did random sampling on the small class to turn it into a rare class, then combined it with the original large class to form a sample data set. Detailed information of the samples can be found in Table 3. We did sampling ten times for each data set and returned the average classification results for it, as shown in Table 4. Finally, since SVMs shows best classification performance in many cases [8], we used it as the benchmark classifier for all the experiments in this subsection.

Results by COG-OS on Two-class Data Sets.

Table 4 shows the performance of COG-OS(SVMs) and pure SVM on six two-class data sets. As can be seen, pure SVM assigned all the instances to the large class of data sets `pima-diabetes`, `fourclass` and `german.numer`. This indicates that pure SVM has no prediction power on rare classes for these three data sets. In contrast, COG-OS can successfully identify more than 25 percent instances of the rare classes. Indeed, the F-measure values of the rare classes by COG-OS(SVMs) are consistently higher than the F-measure values produced by pure SVM for all six data sets, as indicated in Table 4. For instance, for data sets `SVMguide1` and

Table 4: Classification Results of Sampled Data Sets by COG-OS (SVMs).

Data Set	Method	Class	#repetitions	#clusters	Recall	Precision	F-measure
breast-w	SVMs	1	N/A	N/A	0.871	0.878	*0.872*
		2	N/A	N/A	0.986	0.987	0.985
	COG-OS	1	3	1	0.907	0.850	*0.873*
		2	1	4	0.982	0.990	0.986
pima-diabetes	SVMs	1	N/A	N/A	0.000	#DIV/0!	N/A
		2	N/A	N/A	1.000	0.904	0.949
	COG-OS	1	2	1	0.344	0.428	*0.373*
		2	1	4	0.949	0.931	0.940
fourclass	SVMs	1	N/A	N/A	0.000	#DIV/0!	N/A
		2	N/A	N/A	1.000	0.902	0.948
	COG-OS	1	2	1	0.606	0.699	*0.646*
		2	1	4	0.971	0.958	0.964
german.numer	SVMs	1	N/A	N/A	0.000	#DIV/0!	N/A
		2	N/A	N/A	1.000	0.921	0.959
	COG-OS	1	3	1	0.317	0.239	*0.270*
		2	1	4	0.915	0.940	0.927
splice	SVMs	1	N/A	N/A	0.314	0.289	*0.298*
		2	N/A	N/A	0.929	0.938	0.933
	COG-OS	1	3	1	0.493	0.270	*0.344*
		2	1	4	0.874	0.950	0.910
SVMguide1	SVMs	1	N/A	N/A	0.069	0.261	*0.104*
		2	N/A	N/A	1.000	0.976	0.988
	COG-OS	1	9	1	0.913	0.365	*0.518*
		2	1	4	0.956	0.998	0.976

Notes:1.For SVMs: -t 0.
2."#repetitions" means the replicative time of each instance during OS.
3."#clusters" means the preset cluster numbers for K-means during COG.
4."N/A": not applicable; "#DIV/0!": divided by zero.

`fourclass`, COG-OS(SVMs) results in the increases of the F-measure values by more than 0.4.

Table 5: Information of `SVMguide1` Samples.

Sample ID	1	2	3	4	5
Sampling Ratio	0.03	0.05	0.07	0.09	0.11
Size of Rare Class	33	55	77	99	120
#clusters for Larger Class	4	4	4	4	4
#repetitions for Rare Class	15	9	6	5	4

Note:Class sizes of the original data are 1089 and 2000.

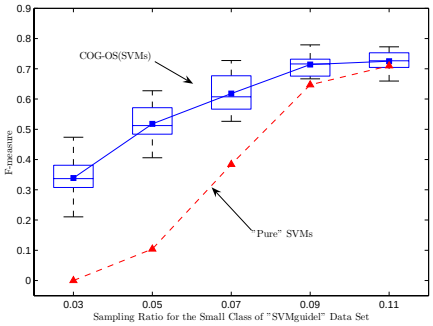

Figure 4: The Effect of the Size of the Rare Class.

Another observation is that, for COG-OS, the increase of the F-measure value of the rare class is **NOT** at the high cost of the prediction accuracy of the large class. For example, for data sets `breast-w` and `fourclass`, the F-measure values of both large and rare classes by COG-OS are higher than the F-measure values produced by pure SVM. Also, for the rest four data sets, the F-measure values of the large class by COG-OS are just slightly smaller. This is acceptable since the rare class is usually the major concern in many real-world applications.

In addition, we also investigate how the F-measure value changes as the increase of the sampling ratio on the small class. As an example on the `SVMguide1` data set, Table 5 shows the information of various samples as the increase of the size of the rare class. Figure 4 shows the classification results on these samples. Please note that for each sampling ratio, we did sampling ten times and therefore had ten samples for each ratio. Then we applied pure SVM and COG-OS on the ten samples respectively, and finally got ten results for each sampling ratio, as indicated by the box plots in Figure 4. As can be seen, the F-measure values by COG-OS are consistently higher than the ones produced by pure SVM, no matter what the sampling ratio is. Moreover, COG-OS performs much better than pure SVM when the rare class size is relatively small. However, as the increase of the size of the rare class, the performance difference is decreased.

Results by COG on Multi-classes Data Sets. In addition to the experiments on the two-class data sets, we also employed some multi-classes data sets with imbalanced classes to validate the COG method. For these multi-classes data sets, we simply used the COG scheme. Since the Euclidean distance is not very meaningful for the sparse doc-

Table 7: COG-OS vs. the Resampling Scheme.

Data Set	Class	COG-OS	US	OS
breast-w	1(rare)	0.873	0.858	0.834
	2	0.986	0.983	0.981
	total	0.975	0.970	0.967
pima-diabetes	1(rare)	0.373	0.320	0.331
	2	0.940	0.818	0.828
	total	0.890	0.714	0.727
fourclass	1(rare)	0.646	0.339	0.346
	2	0.964	0.804	0.809
	total	0.935	0.699	0.704
german.num	1(rare)	0.270	0.255	0.258
	2	0.927	0.812	0.835
	total	0.868	0.701	0.731
splice	1(rare)	0.344	0.268	0.349
	2	0.910	0.805	0.905
	total	0.843	0.694	0.835
SVMguide1	1(rare)	0.518	0.308	0.397
	2	0.976	0.937	0.959
	total	0.955	0.884	0.923

Note: 1.Performances are measured by F-measure.
2.SVMs was used as the only classifier.

Table 6: Classification Results of Multi-classes Data Sets by COG (SVMs).

page-blocks (CV=1.953)		Pure SVM				COG(SVMs)			
Class	#instances	#clusters	Recall	Precision	F-measure	#clusters	Recall	Precision	F-measure
1	4913	N/A	0.990	0.971	0.980	10	0.993	0.977	0.985
2	329	N/A	0.780	0.839	0.808	2	0.880	0.898	0.889
3	28	N/A	0.556	1.000	0.714	1	0.556	0.833	0.667
4	88	N/A	0.815	0.917	0.863	1	0.815	0.917	0.863
5	115	N/A	0.514	0.900	0.655	1	0.543	0.950	0.691
total	5473	N/A	0.962	0.962	0.962	-	0.971	0.971	0.971
k1b (CV=1.316)		Pure SVM				COG(SVMs)			
Class	#instances	#clusters	Recall	Precision	F-measure	#clusters	Recall	Precision	F-measure
1	494	N/A	0.968	1.000	0.984	3	0.974	1.000	0.987
2	1389	N/A	0.990	0.963	0.976	9	0.993	0.981	0.987
3	142	N/A	0.979	0.939	0.958	1	1.000	0.940	0.969
4	114	N/A	0.829	0.967	0.892	1	0.886	0.969	0.925
5	60	N/A	0.810	1.000	0.895	1	0.905	1.000	0.950
6	141	N/A	0.955	0.955	0.955	1	1.000	0.978	0.989
total	2340	N/A	0.969	0.969	0.969	-	0.982	0.982	0.982
la12 (CV=0.503)		Pure SVM				COG(SVMs)			
Class	#instances	#clusters	Recall	Precision	F-measure	#clusters	Recall	Precision	F-measure
1	1848	N/A	0.921	0.561	0.697	2	0.886	0.572	0.695
2	1497	N/A	0.757	0.912	0.827	2	0.708	0.922	0.801
3	1042	N/A	0.663	0.759	0.707	1	0.721	0.728	0.725
4	729	N/A	0.465	0.896	0.612	1	0.564	0.866	0.683
5	642	N/A	0.672	0.794	0.728	1	0.687	0.793	0.736
6	521	N/A	0.329	0.917	0.485	1	0.353	0.881	0.504
total	6279	N/A	0.709	0.709	0.709	-	0.713	0.713	0.713

Table 8: Information of kddcup99data and the Modified Data Sets.

Dataset	Source	#objects	#features	#classes	MinClassZize	MaxClassSize	CV_0
kddcup99data	UCI KDD	494021/292300	41	5	52/39	391458/223298	1.708/1.634
probe_binary†	UCI KDD	494021/292300	41	2	4107/2377	489914/289923	1.391/1.391
r2l_binary†	UCI KDD	494021/292300	41	2	1126/5993	492895/286307	1.408/1.356

Note: We deleted in the test set 18729 instances whose class labels are not present in the training set.
†: Modified binary data set of kddcup99data.

ument data sets k1b and la12, for the COG method, we used the CLUTO [19] implementation of K-means on these two data sets with cosine similarity as the proximity measure. Table 6 shows the classification results by pure SVM and COG. As can be seen, for data set k1b, the F-measure value for every class using COG is higher than that produced by pure SVM. Meanwhile, the results on the page-blocks and la12 data sets show a similar trend as k1b. In summary, COG indeed can improve the prediction performance on rare classes, and this improvement is achieved without a big loss of the prediction performance on large classes.

4.3 COG-OS vs. the Resampling Schemes

In previous sections, we mentioned that resampling is a widely used technique to improve the classification performance on imbalanced data sets. Here, we compare the performances of COG-OS with two resampling strategies — under-sampling and over-sampling. Details of these two resampling methods can be found in various books [29, 23].

In this experiment, we also employed the six sampled data sets as shown in Table 3. We set the sampling ratio for under-sampling or over-sampling to make the modified size of the rare class be approximate to the one of the large class, and the classifier we used here is SVMs. Table 7 shows the results. One observation is that, for all data sets in Table 7, COG-OS performs the best for the rare class, except for one data set: splice, on which COG-OS and over-sampling show comparable results. Another observation is even more encouraging; that is, while obtaining excellent performances on the rare classes, COG-OS also provides much higher predictive accuracies on the large classes, which has long been the "choke point" of the resampling schemes. This is not surprising since the non-linear-separable concept in the data is usually the bottle-neck of the class imbalance problem. By integrating the over-sampling scheme, COG can further improve its ability to identify the instances from the rare class.

In summary, compared to two widely used resampling strategies, COG-OS shows appealing performances on handling non-linear-separable data with rare classes, yet keeps a much better performance on large classes.

4.4 COG-OS for Network Intrusion Detection

Here, we demonstrate an application of COG-OS for network intrusion detection. For this experiment, we used a real-world network intrusion data set, which is provided as part of the KDD-CUP-99 classifier learning contest [3], and now is a benchmark data set in the UCI KDD Archive [4].

The KDD CUP Data Set. The data set was collected by monitoring a real-life military computer network that was intentionally peppered with various attacks that hackers would use to break in. Original training set has close to 5 million records belonging to 22 subclasses and 4 classes of attacks, i.e., dos, probe, r2l and u2r, and still one normal class. In this experiment, we applied 10% sample of this original set which is also supplied as part of the contest. We present results for two rare classes: probe and r2l, whose populations in the 10% sample training set are 0.83% and 0.23%, respectively. The test set provided with the 10% training set, however, has some new subclasses that are not present in the training data. So for the evaluation concern we deleted these new subclasses, and the resultant populations of probe and r2l in the test set are 0.81% and 2.05%, respectively. Table 8 shows the detailed information of these data sets. Note that we obtained the probe_binary

data set by making the probe class as the rare class, and the rest four classes as one large class. The other data set, i.e., r2l_binary, was prepared in a similar fashion.

Table 9: Results on Modified Data Sets.

probe_binary	SVMs	RIPPER	PNrule	*COG-OS*
rare class	0.806	0.798	0.884	*0.881*
huge class	0.998	0.998	N/A	*0.999*
total	0.996	0.996	N/A	*0.998*
r2l_binary	SVMs	RIPPER	PNrule	*COG-OS*
rare class	0.262	0.360	0.230	*0.496*
huge class	0.991	0.992	N/A	*0.993*
total	0.983	0.984	N/A	*0.986*

Note: "N/A" means results were not provided by the source paper [17].

Table 10: Classification Accuracies by COG with Different Classifiers.

Data Set	#clusters	Classifier			
		SVMs	BMR	C4.5	RIPPER
letter	N/A	*0.851*	*0.750*	*0.862*	*0.839*
	2	0.872	0.762	0.846	0.821
	4	0.923	0.812	0.858	0.826
	6	0.946	0.844	0.854	0.818
	8	0.952	0.850	0.856	0.812
optdigits	N/A	*0.965*	*0.949*	*0.858*	*0.874*
	2	0.973	0.958	0.880	0.840
	4	0.973	0.970	0.855	0.816
	6	0.979	0.972	0.866	0.805
	8	0.981	0.973	0.860	0.771
pendigits	N/A	*0.953*	*0.901*	*0.921*	*0.925*
	2	0.969	0.937	0.920	0.918
	4	0.979	0.964	0.927	0.917
	6	0.981	0.962	0.923	0.897
	8	0.977	0.967	0.920	0.867
satimage	N/A	*0.852*	*0.834*	*0.854*	*0.854*
	2	0.860	0.837	0.841	0.855
	4	0.871	0.841	0.857	0.850
	6	0.883	0.862	0.850	0.848
	8	0.881	0.863	0.847	0.847
vowel	N/A	*0.517*	*0.448*	*0.517*	*0.468*
	2	0.602	0.517	0.392	0.312
	4	0.582	0.541	0.385	0.370
	6	0.580	0.491	0.370	0.314
	8	0.597	0.513	0.346	0.251

Notes:
1.All classifiers used default settings except for SVMs: -t 0.
2.For K-means, maxIteration=500, repeat=10.

The Benchmark Classifiers. In this experiment, we apply four classifiers: COG-OS(SVMs), pure SVM, RIPPER [7], and PNrule [17]. For COG-OS, the cluster number for the large class is 4 for each data set, and the replicative times of over-sampling on the rare classes for probe_binary and r2l_binary are 30 and 120, respectively. For SVMs, we set the parameters as: -t 0. Ripper and PNrule are two rule induction classifiers. RIPPER builds rules first for the smallest class and will not build rules for the largest class. Hence, one might expect that RIPPER can provide a good performance on the rare class. As to PNrule, it consists of positive rules (P-rules) that predict presence of the class, and negative rules (N-rules) that predict absence of the class. It is right the existence of N-rules that can ease the two problems induced by the rare class: splintered false positives and error-prone small disjuncts. These two classifiers have shown the appealing performance on classifying the modified binary data sets in Table 8, and the PNrule classifier even shows superior performance [17]. To our best knowledge, we used the same source data as [17] and the pre-process procedure for the modified data sets is also very similar to [17]. Therefore, we simply adopted the results of PNrule in [17] for our paper.

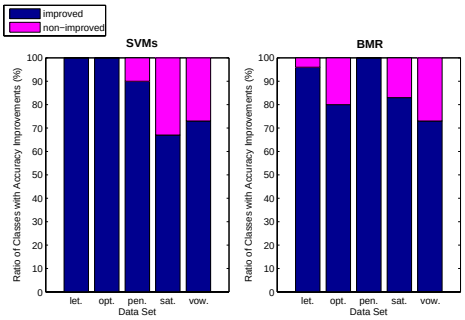

Figure 5: Ratio of the Classes with Accuracy Improvements by COG.

The Results. Table 9 shows the classification results by various methods on the probe_binary data set. As can be seen, COG-OS performs much better than pure SVM and RIPPER on predicting the rare class as well as the large class, while PNrule shows slightly higher F-measure on the rare class. For data set r2l_binary, however, COG-OS shows overwhelming advantages among all classifiers. As indicated in Table 9, the F-measure value of the rare class by COG-OS is 0.496, far more higher than the ones produced by the rest classifiers. Meanwhile, the predictive accuracy of the large class by COG-OS is also higher than that of pure SVM and RIPPER. This real-world application nicely illustrates the effectiveness of COG-OS — the combination of local clustering and over-sampling schemes. We believe that COG-OS is a prospective solution to the difficult classification problem induced by the non-linear-separable concepts and imbalanced class distributions.

4.5 The Effect of COG on Balanced Data Sets

In the previous subsections, we have shown that COG and COG-OS indeed can improve the prediction accuracies of the rare classes for imbalanced data sets. In this subsection, however, we would like to show COG is also applicable to data sets with balanced class distributions.

In this experiment, we used five balanced data sets with CV < 0.5, i.e., letter, optdigits, pendigits, satimage and vowel. Among them, four data sets have been split into training and test sets by UCI repository except for the letter data set. Four classifiers including SVMs, BMR, C4.5 and RIPPER were used for the purpose of comparison. The clustering method in the COG scheme is K-means with Euclidean notion proximity, and the cluster number for each class in a data set is exactly the same, ranging from 2 to 8.

Results by COG with Linear Classifiers. Table 10 shows the experimental results on these balanced data sets. As can be seen, for linear classifiers SVMs and BMR, COG indeed can improve the classification accuracies no matter what the cluster number is. For instance, for the data set letter, the accuracies achieved by pure SVM and BMR are merely 0.851 and 0.750 respectively (as indicated by the italic numbers). In contrast, COG with SVMs and BMR can increase the prediction accuracies of the rare classes steadily as the increase of the cluster number, and finally up to 0.952 and 0.850 respectively when the cluster number is 8. Indeed, the resultant prediction accuracies are 10% higher than the ones obtained by pure SVM and BMR.

Next, we take a closer look at the performance of COG in the class-wise level. Table 11 shows the classification accuracies on the data set optdigits by pure SVM and COG.

Table 11: Classification Accuracies of `optdigits` in the Class-wise Level.

Class	1	2	3	4	5	6	7	8	9	10
SVMs	0.994	0.967	0.960	0.934	0.989	0.989	0.989	0.950	0.920	0.956
COG(SVMs)	1.000	0.989	0.994	0.973	1.000	0.995	0.994	0.950	0.954	0.956
BMR	0.972	0.940	0.977	0.918	0.972	0.978	0.978	0.911	0.902	0.939
COG(BMR)	1.000	0.978	0.989	0.967	0.967	0.973	0.989	0.961	0.948	0.961

Note: For COG, the cluster number is set to be 8 for every class.

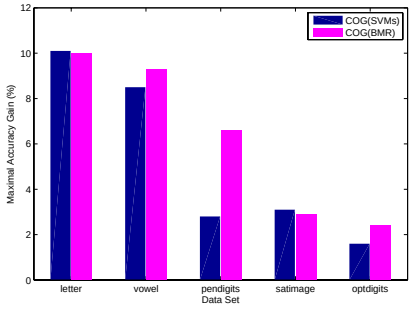

Figure 6: Comparison of the Maximal Accuracy Gains by COG with SVMs and BMR.

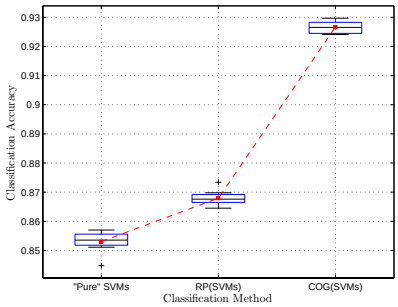

Figure 7: Comparison of the Classification Accuracies by COG and Random Partitioning.

In the table, we can observe that COG can simultaneously improve the classification accuracies for nearly all the classes of `optdigits`. In addition, Figure 5 shows the improvement ratio of classes in the five balanced data sets by COG with SVMs and BMR ("#clusters"=8). A very similar improvement trend can be observed for all five data sets. Indeed, COG can transform the non-linear-separable concepts in the data into the linear-separable concepts so as to simultaneously improve the classification performance of linear classifiers for most of the classes in the data.

Another interesting observation is that, the accuracy improvements gained by COG with SVMs and BMR are quite close. To illustrate this, we compute the *maximal accuracy gain* for each data set; that is, we first select the highest accuracy among four values achieved in different "#clusters" levels, then subtract it by the accuracy obtained by the pure classifier. Figure 6 shows the results. As can be seen, the maximal accuracy gains of all data sets by COG with SVMs and BMR are quite close except for `pendigits`. This implies that the non-linear-separable concept in the data is the bottle-neck that hinders the analysis of linear classifiers.

Results by COG with Non-linear Classifiers. Table 10 also shows the results of COG with non-linear classifiers such as RIPPER and C4.5 on five balanced data sets. In the table, we can see that COG(RIPPER) has worse performance than pure RIPPER on all five data sets. This is due to the fact that the rule learning algorithm aims to build up a rule set in a greedy fashion by employing the standard divide-and-conquer strategy. Meanwhile, COG partitions instances of the same class into different sub-classes. This can increase the number of negative examples for some target rules, and ultimately result in missing such rules.

Finally, for another widely used non-linear classifier C4.5, the performance of COG with C4.5 is not consistent on five balanced data sets as shown in Table 10. For instance, COG can improve the classification accuracy of `optdigits`, but lead to worse performances on `letter`, `vowel` and `satimage`. This is due to the fact that COG can increase the number of sub-classes so as to make the branch-splitting decision even harder to make. In other words, the splitting attributes of the tree can be better or worse selected in such "uncertain" scenarios, which results in the inconsistent performances.

COG versus Random Partitioning. In this experiment, we compare the effect of clustering in the COG scheme with that of simple random partitioning. To this end, we take the data set `letter` and the classifier SVMs to illustrate this. First, we randomly split the `letter` data set into two parts, 70% of which as the training set and the rest as the test set (the class size distribution holds). Then we performed training and testing in a very similar fashion to the procedure of COG except that we use random partitioning instead of clustering on each class. Figure 7 shows all the results. Please note that for random partitioning and COG, "#clusters"= 4. As indicated by Figure 7, while the performance of random partitioning with SVMs, i.e., RP(SVMs), are slightly better than the results of pure SVM, they are much worse than the results by COG(SVMs). This indicates that the clustering phase in COG is very important. In fact, the local clustering process can transform the non-linear-separable concepts of the data into linear or "quasi" linear concepts, and so as to improve the classification accuracy of linear classifiers.

In summary, COG is of great use on improving the classification accuracy of linear classifiers by eliminating or mitigating the negative impact of the non-linear-separable concepts in the data. But for the non-linear classifiers, such as C4.5 and RIPPER, COG shows no competitive results.

5. RELATED WORK

In the literature, there are a number of methods addressing the class imbalance problem. For instance, the sampling based methods are one of the simplest yet effective ones. The over-sampling scheme replicates the small classes to match the sizes of large classes [22, 29]; under-sampling, however, cuts down the large class sizes to achieve a similar effect [21, 29]. Drummond and Holte [11] provided detailed comparisons on these two resampling schemes. Another popular method is the cost-sensitive learning scheme which takes the cost matrix into consideration during model building and generates a model that has the lowest cost. The properties of a cost matrix had been studied by Elkan [13]. Margineantu and Dietterich [24] examined various methods for incorporating cost information into the C4.5 learning

algorithm. Other cost-sensitive learning methods that are algorithm-independent include AdaCost [14], MetaCost [10], and Costing [32]. In addition, Joshi et al. [18] discussed the limitations of boosting algorithms for rare class modeling and proposed PNrule, a two-phase rule induction algorithm, to handle the rare class purposefully [17]. Other algorithms developed for mining rare classes include SMOTE [6], RIPPER [7] etc. A survey paper is given by Weiss [31].

Finally, in her inspiring paper, Japkowicz [16] shows the idea of "supervised learning with unsupervised output separation". This work shares some common grounds with our COG method in terms of combining supervised and unsupervised learning techniques. However, in this paper, we have a novel perspective on rare class analysis. We develop the foundation of classification using local clustering (COG) for enhancing linear classifies on handling both balanced and imbalanced classification problems.

6. CONCLUSIONS

In this paper, we propose a method for classification using local clustering (COG). The key idea is to perform clustering within each class and produce linearly separable subclasses with relatively balanced sizes. For data sets with imbalanced class distributions, the COG method can improve the performance of traditional supervised learning algorithms, such as Support Vector Machines (SVMs), on rare class analysis. In addition, the COG method has the capability in enhancing linear classifiers on data sets containing non-linear-separable classes. Finally, as demonstrated by our experimental results on various real-world data sets, COG with over-sampling can have much better prediction performance on rare classes than state-of-the-art methods.

7. ACKNOWLEDGEMENTS

This research was partially supported by the National Science Foundation of China (NSFC) Research Fund Nos. 70621061 and 70518002. Also, this research was supported in part by a Faculty Research Grant from Rutgers Business School-Newark and New Brunswick.

8. REFERENCES

[1] Bmr. In *http://www.stat.rutgers.edu/ madigan/BMR/*.

[2] C4.5. In *http://www.rulequest.com/Personal/*.

[3] Kddcup. In *http://www.acm.org/sigs/sigkdd/kddcup/index.php*.

[4] Kddcup99data. In *http://kdd.ics.uci.edu//databases/kddcup99/kddcup99.html*.

[5] Libsvm. In *www.csie.ntu.edu.tw/ cjlin/libsvm/*.

[6] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of AI Research*, 16:321–357, 2002.

[7] W. Cohen. Fast effective rule induction. In *ICML*, pages 115–123, 1995.

[8] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. Cambridge University Press, 2000.

[9] M. DeGroot and M. Schervish. *Probability and Statistics (3 edition)*. Addison Wesley, 2001.

[10] P. Domingos. Metacost: a general method for making classifiers cost-sensitive. In *KDD*, pages 155–164, 1999.

[11] C. Drummond and R. Holte. C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. In *ICML Workshop*, 2003.

[12] R. Duda, P. Hart, and D. Stork. *Pattern classification*. Wiley New York, 2001.

[13] C. Elkan. The foundations of cost-sensitive learning. In *IJCAI*, pages 973–978, 2001.

[14] W. Fan, S. Stolfo, J. Zhang, and P. Chan. Adacost: misclassification cost-sensitive boosting. In *ICML*, pages 97–105, 1999.

[15] E.-H. Han and et al. Webace: A web agent for document categorization and exploration. In *Int'l Conf. on Autonomous Agents*, 1998.

[16] N. Japkowicz. Supervised learning with unsupervised output separation. In *Int'l Conf on Artificial Intelligence and Soft Computing*, pages 321–325, 2002.

[17] M. Joshi, R. Agarwal, and V. Kumar. Mining needle in a haystack: Classifying rare classes via two-phase rule induction. In *SIGMOD*, pages 91–102, 2001.

[18] M. Joshi, R. Agarwal, and V. Kumar. Predicting rare classes: Can boosting make any weak learner strong? In *KDD*, 2002.

[19] G. Karypis. Cluto – software for clustering high-dimensional datasets, version 2.1.1. In *http://glaros.dtc.umn.edu/gkhome/views/cluto*.

[20] M. Kubat, R. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar imaages. *Machine Learning*, 30:195–215, 1998.

[21] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *ICML*, pages 179–186, 1997.

[22] C. Ling and C. Li. Data mining for direct marketing: Problems and solutions. In *KDD*, pages 73–79, 1998.

[23] O. Maimon and L. Rokach, editors. *The Data Mining and Knowledge Discovery Handbook*. Springer, 2005.

[24] D. Margineantu and T. Dietterich. Learning decision trees for loss minimization in multi-class problems. In *TR 99-30-03*. Oregon State University, 1999.

[25] P. Murphy and D. Aha. In *UCI Repository of Machine Learning Databases*. U. of California at Irvine, 1994.

[26] D. Newman, S. Hettich, C. Blake, and C. Merz. Uci repository of machine learning databases, 1998.

[27] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[28] S. Raudys and A. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *TPAMI*, 13(3):252–264, 1991.

[29] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[30] TREC. In *http://trec.nist.gov*.

[31] G. Weiss. Mining with rarity: a unifying framework. *ACM SIGKDD Explorations*, 6(1):7–19, 2004.

[32] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *ICDM*, pages 435–442, 2003.

[33] J. Zurada, B. Foster, and T. Ward. Investigation of artificial neural networks for classifying levels of financial distress of firms: The case of an unbalanced training sample. In *Knowledge Discovery for Business Information Systems*, pages 397–423. Kluwer, 2001.