Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Modeling Skewed Class Distributions by Reshaping the Concept Space

Kyle D. Feuz
School of Computing
Weber State University

Diane J. Cook
School of Electrical Engineering and Computer Science
Washington State University

Abstract

We introduce an approach to learning from imbalanced class distributions that does not change the underlying data distribution. The ICC algorithm decomposes majority classes into smaller sub-classes that create a more balanced class distribution. In this paper, we explain how ICC can not only address the class imbalance problem but may also increase the expressive power of the hypothesis space. We validate ICC and analyze alternative decomposition methods on well-known machine learning datasets as well as new problems in pervasive computing. Our results indicate that ICC performs as well or better than existing approaches to handling class imbalance.

Introduction

Skewed class distributions present a challenge in many different domains. Specifically, most supervised machine learning algorithms exhibit poor performance when faced with skewed class distributions. This is referred to as the class imbalance problem.

Class imbalance often occurs in real-life datasets involving rare events such as detecting certain medical conditions (Mazurowski et al. 2008), fraudulent transactions (Bhattacharyya et al. 2011), or providing prompts in context-aware situations (Feuz et al. 2015). Additionally, class imbalance occurs on datasets with several classes of all different sizes such as location prediction or activity recognition (Feuz and Cook 2015).

Several approaches have been developed to address class imbalance including sampling, re-weighting the instances, applying cost-sensitive learning or developing specialized learning algorithms. Sampling techniques can be used in conjunction with any learning algorithm because they only alter what information is provided to the learning algorithm. This is convenient for the learning algorithm but can lead to duplicated information, removal of important information, or creation of false information. The last three approaches do not alter the training instances and are not affected by these problems but instead require altering the learning algorithm itself.

Intra-class clustering (ICC) uses clustering to decompose a large majority class into smaller sub-classes leading to a more balanced distribution. One benefit of applying ICC is that it can also help solve other issues that arise in supervised machine learning. For example, several researchers have found that class imbalance is especially problematic when there exist different sub-clusters within the classes (Stefanowski 2013). ICC can be used to identify these sub-clusters and to model them as new distinct classes. Additionally, by creating more sub-classes, we increase the expressive power of certain hypothesis spaces. As an example, when using a standard support vector machine (SVM(without employing a specialized kernel, SVMs can only correctly classify linearly separable classes. By creating sub-classes through clustering, the likelihood of a class to be linearly separable increases with the increase in the number of decision boundaries. Increasing the expressive power of the hypothesis space may lead to over-fitting or increased runtimes and does not guarantee better results.

We hypothesize that decomposing the majority class(es) into smaller sub-classes prior to training a classifier will 1) lead to improved performance of the learned model by 2) creating a more balanced class distribution, 3) creating the potential for more decision boundaries, and 4) finding intrinsic sub-class boundaries. We validate this novel approach using a set of existing machine learning datasets as well as two new pervasive computing datasets, all with extreme class imbalance.

Methods

ICC is applied as a pre-processing step prior to learning a classification model. First, a class or classes are individually decomposed into sub-classes. The training instances are assigned new class labels corresponding to their respective sub-class. For example, a class c_i might be decomposed into three sub-classes c_{i1}, c_{i2}, and c_{i3} and the corresponding training data is mapped to the new subclass labels. This training data is then used to build a classification model according to the standard supervised machine learning procedure. New instances can now be classified using the trained classification model. Note that the classifier will predict class labels using the discovered sub-class labels. However, these new labels can be converted back into the original class labels. Figure 1 outlines the entire process.

More formally, we use the following notation throughout rest of this paper.

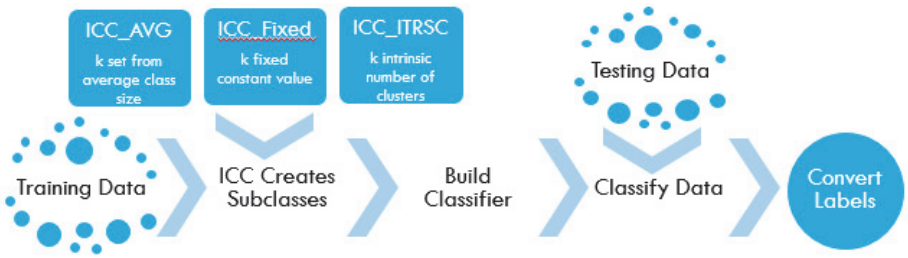

Figure 1: Overview of the intra-class clustering process

- $X = \{(x_1, y_1), (x_2, y_2)...(x_n, y_n)\}$: the set of labeled training instances
- $x_i =< a_1, a_2, a_m >$: the ith training instance consisting of m attributes
- $C = \{c_1, c_2, ...c_p\}$: the set of possible class labels
- $y_i \in C$: the class label for the ith training instance
- $X' = \{x'_1, x'_2, ...x'_{n'}\}$: the set of unlabeled testing instances
- $X_{ci} = \{(x, y) \in X | y = c_i\}$: the set of training instances with class label c_i
- $|X_{ci}|$: the number of instances in X_{ci}

In traditional supervised machine learning a set of training instances X is fed to a learning algorithm which learns a function $f : X \rightarrow C$. This function can be applied to the unlabeled testing instances to predict label y'_i for instance x'_i.

In intra-class clustering, C is first expanded by decomposing one label c_a into sub-labels $S_{ca} = \{c_{a1}, ...c_{ak}\}$. This leads to an expanded set of class labels, $C' = \{c_1, ...c_{a-1}, c_{a1}, ...c_{ak}, c_{a+1}, ...c_p\}$. ICC also generates a function $g : X_{ca} \rightarrow S_{ca}$ which maps training instances with the original label c_a to a new sub-label drawn from S_{ca}. ICC creates this mapping by clustering the instances of X_{ca} into sub-classes c_{a1} through c_{ak}. The training instances X are updated to $X_h = \{(x, h(y)) | (x, y) \in X\}$ which essentially replaces the label y with the appropriate sub-class label.

$$h(y) = \begin{cases} g(y) & \text{if } y = c_a \\ y & \text{if } y \neq c_a \end{cases} \quad (1)$$

While the goal of the original classifier was to learn a function $f : X \rightarrow C$, now the classifier learns a mapping to an expanded set of class labels, or $f : X \rightarrow C'$. For applications requiring only the original set of labels C, the sub-labels need to be converted back to their parent class using function e, where

$$e(f(x)) = \begin{cases} c_a & \text{if } f(x) \in S_{ca} \\ f(x) & \text{if } f(x) \notin S_{ca} \end{cases} \quad (2)$$

Before applying intra-class clustering we need to 1) decide which labels will be decomposed, 2) determine how many clusters or sub-classes will be formed, and 3) select the clustering algorithm to use. We consider three different ways of selecting labels to decompose: ICC_One, ICC_Maj, and ICC_All. ICC_One builds on the principal that the largest majority class is the one causing the most imbalance. ICC_One, therefore, selects the class with the greatest proportion number of training instances to divide into sub-classes. Using this method we let $c_a = \arg\max_{c_i \in C} |X_{ci}|$.

In multiclass learning problems, there may be more than one majority class that needs to be decomposed to obtain a more uniform class distribution. For these cases, ICC_Maj selects a set of the largest classes. We now update the previous equations and formulations. Let

$$I = \left\{ c_i \in C | \frac{|X_{ci}|}{\frac{1}{p}\sum_{j=1}^{p} |X_{cj}|} > 1.5 \right\} \quad (3)$$

be the set of class labels which have more instances than 1.5 times the average class size. Let $C' = (C - c) \cup S_c$ for all $c \in I$ be the set of new class labels including sub-labels. Let $g_i : X_{ci} \rightarrow S_{ci}$ be the functions mapping instances of class label c_i to sub-labels S_{ci}. Equation 1 is re-written as

$$h(y) = \begin{cases} g_i(y) & \text{if } y \in I \\ y & \text{else } y \notin I \end{cases} \quad (4)$$

Equation 2 is re-written as

$$e(f(x)) = \begin{cases} c_i & \text{if } \exists c_i \in I | f(x) \in S_{ci} \\ f(x) & \text{if } \nexists c_i \in I | f(x) \in S_{ci} \end{cases} \quad (5)$$

Using these updated equations we can apply ICC as outlined previously.

In addition to creating more balanced sub-classes, ICC can also be used for identifying intrinsic sub-classes. These sub-classes may be present in either the majority classes or the minority classes. Therefore, ICC_All creates sub-classes for all of the class labels. Using the same formulation as ICC_Maj, let $I = C$ for ICC_All. ICC_All decomposes all of the original classes regardless of which class labels are majority classes and which class labels are minority classes.

Additionally, we have three different techniques for determining the number of clusters. The first technique, ICC_Avg, seeks to achieve a more balanced class distribution by calculating the number of clusters needed to decompose the class into sub-classes of average size. The number of sub-classes created is calculated as the size of the class divided by the average class size (see Eq. 6).

$$NumOfClusters = [\frac{|X_{ci}|}{\frac{1}{p}\sum_{j=1}^{p} |X_{cj}|}] \quad (6)$$

The second technique, ICC_Fix, uses a fixed number of clusters per class. This can be useful when a domain expert has knowledge about the classes and knows that each class is really composed of x sub-classes. This can also be useful for our third hypothesis when the goal is to create more decision boundaries for the learning algorithm.

The third technique, ICC_X (where X changes based on the clustering algorithm), addresses our fourth hypothesis when the classes contain an unknown number of intrinsic sub-classes. In this case, we can use a clustering algorithm which does not require that the number of clusters be specified a priori. Instead, the clustering algorithm itself determines how many clusters to create from the training data itself.

ICC can be used with virtually any clustering algorithm. However, some clustering algorithms make more sense than others for a given problem. If the decomposition goal is to achieve a more balanced class distribution, then a clustering algorithm for which the number of clusters can be set implicitly should be used. Furthermore, a clustering algorithm which seeks to achieve clusters of equal size will lead to a more balanced class distribution. If on the other hand, the goal is to decompose a class into an unknown number of intrinsic sub-classes, then it makes sense to choose a clustering algorithm which determines the number of clusters automatically. We evaluate ICC using three different clustering algorithms: k-means++ (Arthur and Vassilvitskii 2007), Expectation Maximization Clustering, and CascadeSimpleKMeans (Caliński and Harabasz 1974). k-means++ must be given the number of clusters while EM clustering and CascadeSimpleKMeans can both determine the number of clusters algorithmically. Our source code and binary jar files are available[1] as Weka add-on packages.

Metrics

Because there are multiple considerations when learning from imbalanced class distribution, we evaluate ICC several alternate metrics. Accuracy is a commonly-used metric for classification. However, it can be misleading in the face of skewed class distributions because a classifier that is optimized for this metric will always select the majority class and not effectively learn the smaller classes. The F1-score is sometimes used in these situations because it represents a trade-off between Recall and Precision. Precision, Recall and F1-Scores are calculated on a per class basis.

When evaluating multi-class problems we need to average the scores from each class into one overall score. The averaging can be performed over all of the instances (micro F1-Score) or over all of the classes (macro F1-Score, see Equation 7), depending on how the values are weighted. The micro F1-Score considers each instance of equal importance so classes with more instances are weighted more heavily. The macro F1-Score considers each class of equal importance so all classes are weighted the same regardless of the number of instances per class. In our experiments we consider the accuracy score as well as the macro F1-score.

[1] http://icarus.cs.weber.edu/ kfeuz/weka/

Table 1: Statistics and characteristics of the datasets

Dataset	# of classes	Maj. Class %	Min. Class %	KL-Div.
abalone	9	39.5	0.2	1.06
car	4	70	3.8	0.79
ecoli	8	40.5	2.3	0.63
glass	6	35.5	4.2	0.41
haberman	2	73.5	26.5	0.17
letters-bin	2	80.6	19.4	0.29
letters-cons	6	80.6	3.8	1.43
letters-vwls	22	19.4	3.6	0.21
nursery	5	33.3	0.06	0.6
yeast4	2	97	3	0.81
yeast6	2	96.6	3.4	0.78
yeast	10	31.2	0.4	0.83
al	16	23	0.2	0.49
al-location	10	40	0.06	1.17
prompting	2	96.3	3.7	0.77

$$macro\ F1 = \frac{1}{p}\sum_{i=1}^{p}\frac{2Recall_i * Precision_i}{Recall_i + Precision_i} \quad (7)$$

In addition to measuring the performance of the classifier, we also need to measure how balanced the datasets are both before and after applying the ICC technique. KL-Divergence is a measure of the divergence between two probability distributions (see Equation 8). We use KL-Divergence to measure how balanced a dataset is by comparing the observed class distribution, P, to a perfectly balanced class distribution (i.e., the uniform class distribution), Q.

$$KLDivergence(P||Q) = \sum_{i=1}^{p} P(i)log\frac{P(i)}{Q(i)} \quad (8)$$

Datasets

We evaluate ICC using fifteen different datasets. Twelve of the datasets come from the UC-Irvine Machine learning repository (Lichman 2013) and exhibit varying amounts of class imbalance. We merged several classes into one for a few of the datasets to create classification problems with known intrinsic sub-classes. Three of the datasets represent pervasive computing challenges generated by the CASAS project. Class distribution statistics for each dataset are listed in Table 1.

The CASAS datasets center around activity-aware applications. The prompting dataset consists of data gathered in a smart home with 128 volunteer participants, aged 50+, who are healthy older adults or individuals with mild cognitive impairment (Das et al. 2016). The smart home is a two-story apartment equipped with sensors that monitor motion, door open/shut status, and usage of water, burner, and specific items throughout the apartment. Clinically-trained psychologists watch over a web camera as the participants perform

8 different activities. We view automated prompting as a supervised learning problem in which each activity step is mapped to a "Prompt" or "No-prompt" class label.

The al and al-loc datasets have been gathered from 99 volunteers who downloaded and ran our activity learner (AL) mobile application. AL runs on IOS and Android platforms and is publicly available[2]for the community to download and use. AL collects 5 seconds of sensor data at specified intervals and the users are prompted for ground truth labels.

Results

We have designed several experiments that address the original hypotheses. The first experiment is designed to determine whether ICC is an effective technique to improve classifier performance. We apply ICC in conjunction with four different classification algorithms: Naïve Bayes, J48 Decision Tree, Support Vector Machines One-vs-All (LibLINEAR) (Fan et al. 2008) and Support Vector Machines One-vs-One (LibSVM) (Chang and Lin 2011).

We also compare ICC against three other class imbalance techniques: SMOTE, Resample, and Balance. SMOTE uses nearby instances from the minority class to artificially generate new minority class instances (Chawla et al. 2002). We run SMOTE on every instance of the smallest class, effectively doubling the size of the minority class. SMOTEBoost combines the SMOTE sampling technique qith a boosting algorithm (Chawla et al. 2003). Resample samples with replacement instances from the dataset to produce a balanced class distribution (Hall et al. 2009). Balance assigns weights to each instance so that the total weight for each class is balanced (Hall et al. 2009). These weights only affect classifiers which are able to account for instance weighting.

ICC itself can be run with different configurations. We show the results of choosing the number of clusters based on the average class size (ICC_Avg), fixing the number of clusters to a constant $k = 15$ (ICC_Fix), and choosing the number of clusters algorithmically on a per class basis using CascadeSimpleKMeans (ICC_CSK) or Expectation Maximization (ICC_EM). We also consider applying ICC to all (All) of the classes or only to the majority (Maj) classes (i.e., those classes with more instances than average). We run 10 iterations of 3-fold cross validation for each dataset to determine which results are significant.

Table 2 shows the number of statistically significant differences in accuracy and macro F1-scores when compared to not applying any class imbalance technique. Entries with the highest number of significant improvements and the lowest number of significant performance decreases for a given classifier are indicated with bold font.

In general, the ICC techniques significantly improve classifier performance as often or more often than any of the four comparison algorithms, SMOTE, SMOTEBoost, Resample, or Balance. Only when the J48 classifier is used does SMOTEBoost consistently outperform the ICC techniques. The ICC techniques offer the most improvement when matched with the Naïve Bayes learning algorithm. They also frequently improve the performance of the LibSVM classifier. Only a few of the datasets show improvement when ICC is combined with the LibLINEAR classifier and none of the datasets show significant improvement when combined with the J48 classifier unless SMOTEBoost is also applied. When we consider applying ICC to all of the classes as opposed to only applying ICC to the majority classes, there is no clear difference. All of these results support the first hypothesis that ICC is an effective technique to improve classifier performance on datasets with imbalanced class distributions.

The next experiment explicitly considers the relationship between classifier performance and class distribution. We calculate Pearson's correlation coefficient to measure the linear relationship between each performance metric and the KL-Divergence of the data. Our second hypothesis stated that improved class balance should lead to improved classification results so we expect a negative correlation between the performance measure and the KL-Divergence. Table 3 lists the number of datasets exhibiting a negative correlation. The number of datasets for which the correlations were significant ($p < 0.05$) is shown in parentheses. In most cases, more than 33% of the datasets showed the expected negative correlation with KL-Divergence. These results support our second hypothesis that creating a more balanced class distribution leads to improved classifier performance.

The next experiment analyzes the effect of choosing a fixed number of sub-classes k to create. Ten iterations of 3-fold cross validation using ICC_Fix_All is run on all of the datasets. k is fixed at values of 2, 3, 5, 10 and 15. In most cases, increasing k leads to further improvement in the accuracy. On some datasets, like haberman, increasing k causes the accuracy to decrease. On other datasets, like glass, increasing k improves the accuracy up to a point and then the accuracy decreases again. These results support our third hypothesis, increasing the number of decision boundaries typically leads to an increase in classification performance but eventually may lead to over-fitting and subsequent decrease in performance.

Another way of considering our third hypothesis is to examine the correlation between the total number of classes and classification accuracy. As before, we calculate the Pearson's Correlation Coefficient between the total number of classes and classification accuracy. As shown in Figure 3, approximately half of the datasets exhibit a positive correlation between the number of classes and the classification accuracy when LibSVM or Naïve Bayes is used. In both cases, 4 of the correlations are statistically significant ($p < 0.05$).

Further support for our third hypothesis is found when comparing LibLINEAR and LibSVM. LibLINEAR and LibSVM are both configured as linear SVMs and as such have highly constrained hypothesis spaces. The major difference between LibLINEAR and LibSVM is that LibLINEAR addresses the multi-class problem with a one-vs-all approach. Each time a class is added, one new decision boundary is thus created. LibSVM, on the other hand, uses a one-vs-one approach to the multi-class problem. In this case, adding a new class actually creates k new decision boundaries, one for each of the previous classes and the new class.

[2]http://casas.wsu.edu/tools/

Table 2: Number of datasets (out of 15) with statistically significant ($p = 0.05$) better ($>$) and worse ($<$) performance (accuracy and f-score) when compared to not using any class imbalance technique.

	NB				LibLINEAR				J48				LibSVM			
	Accuracy		F1-Score		Accuracy		F1-Score		Accuracy		F1-Score		Accuracy		F1-Score	
	$>$	$<$	$>$	$<$	$>$	$<$	$>$	$<$	$>$	$<$	$>$	$<$	$>$	$<$	$>$	$<$
SMOTE	2	3	3	**1**	0	**0**	2	**0**	0	**0**	0	**0**	2	**0**	5	**0**
SMOTEBoost	2	5	3	5	2	4	4	3	**7**	1	**7**	**0**	0	4	0	2
Resample	0	10	2	6	0	7	3	**0**	0	11	1	6	0	7	**8**	1
Balance	0	11	2	6	0	**0**	0	**0**	0	10	1	3	0	**0**	0	**0**
ICC_Avg_Maj	6	1	4	**1**	0	2	1	2	0	3	0	2	2	2	3	**0**
ICC_Fix_Maj	8	1	4	2	0	5	3	2	0	5	0	3	6	3	5	**0**
ICC_Fix_All	7	4	5	3	0	4	4	5	0	6	0	6	4	4	3	4
ICC_CSK_Maj	6	2	4	2	0	2	2	2	0	3	0	2	2	2	3	**0**
ICC_CSK_All	4	3	3	2	0	4	2	3	0	4	0	3	2	3	2	2
ICC_EM_Maj	**9**	**0**	6	**1**	1	1	3	**0**	0	5	0	4	**7**	1	5	**0**
ICC_EM_All	**9**	**0**	**8**	**1**	**2**	**0**	**5**	1	0	4	0	4	5	2	7	1

Table 3: Number of datasets exhibiting a linear correlation between the accuracy (Acc) or macro f-score (F), the KL-Divergence (kl), and number of created sub-classes (#sub). #sub generates positive correlations while kl generates negative correlations. Correlations which are statistically significant are indicated by ().

	Acc/ent	Acc/kl	F/ent	F/kl	Acc/#sub
NB	12(7)	12(8)	8(1)	8(4)	8(4)
Linear	8(2)	9(5)	13(3)	10(7)	5(4)
J48	2(1)	9(3)	7(2)	11(5)	1(0)
SVM	8(3)	8(6)	12(5)	10(6)	7(4)

LibSVM is more likely to be improved by using ICC and less likely to see a drop in performance. This supports our third hypothesis that creating more decision boundaries leads to improved performance.

The performance of ICC_EM and ICC_CSK also support our fourth hypothesis that finding intrinsic sub-classes will lead to improved performance. ICC_EM almost always shows the best performance. This is the only technique that improves accuracy for LibLINEAR and it consistently improves performance on more datasets than any other technique. Similarly, in several cases, ICC_CSK also leads to improve performance. ICC_EM and ICC_CSK work by finding intrinsic sub-classes, providing evidence for our fourth hypothesis that doing so leads to improved performance.

Although ICC_EM and ICC_CSK find intrinsic sub-classes, we suspect that representing such sub-classes will also lead to more balanced datasets. Using KL-Divergence we measure how close to uniform the class distribution is. We measure the KL-Divergence of the dataset before and after each ICC technique is applied and calculate the average change in divergence. A negative change in KL-Divergence (shown in Table 4) indicates a more balanced dataset. When applied only to the majority classes, both ICC_EM and ICC_CSK almost always lead to more balanced datasets. When applied to all of the classes, ICC_EM and ICC_CSK will occasionally lead to a more balanced dataset.

Table 4: Average change in KL-divergence after creating the sub-classes. A negative change in KL-Divergence indicates the class distribution is closer to normal.

	ICC_CSK		ICC_EM	
	Maj	All	Maj	All
abalone	-0.73	-0.23	-0.63	-0.54
al	-0.22	0.03	-0.01	0.5
al-location	-0.43	0.41	-0.1	0.32
car	-0.35	0.04	-0.4	-0.25
ecoli	-0.22	0.6	-0.19	-0.09
glass	-0.12	0.98	-0.15	-0.13
haberman	0	0.03	0	0.1
letters-bin	-0.21	0.01	0.13	-0.09
letters-cons	-0.58	0.43	-1.27	-0.85
letters-vwls	-0.13	0.15	-0.05	0.18
nursery	-0.27	-0.02	0.28	0.36
prompting	-0.31	0.14	-0.45	-0.05
yeast4	-0.34	0.6	0.18	0.42
yeast6	-0.3	0.27	0.22	0.49
yeast	-0.39	0.58	-0.05	0.16

The final set of experiments looks for dataset characteristics that lead to strong ICC performance, or conversely, for characteristics of the datasets which lead to ICC doing poorly. Only two of the tested datasets, yeast and ecoli never showed significant improvement for any of the ICC approaches. Two other datasets, haberman and prompting, only showed significant improvement on the LibSVM and NaiveBayes algorithms, respectively. The letter-recognition datasets showed significant improvement the most consistently.

In addition to considering which datasets show significant improvement we also look at the best improvement

Table 5: Best average F1-score for each dataset.

	NB	SVM	Linear
abalone	0.06	0.00	0.03
al	0.05	0.06	0.00
al-location	0.23	0.06	0.01
car	0.23	0.05	0.16
ecoli	0.00	0.07	0.00
glass	0.16	0.13	0.03
haberman	0.02	0.13	0.11
letters-bin	0.18	0.42	0.27
letters-cons	0.27	0.35	0.19
letters-vow	0.28	0.16	0.16
nursery	0.19	0.07	0.21
prompting	0.01	0.07	0.04
yeast	0.00	0.02	0.00
yeast4	0.01	0.22	0.15
yeast6	0.01	0.05	0.12

shown using any ICC technique on all of the datasets. Table 5 shows the results. Across the board, the yeast dataset shows the least amount of improvement. For different learning algorithms, the abalone, ecoli, and haberman datasets also show little improvement. One shared characteristic of these datasets is that they are unlikely to contain intrinsic sub-classes. On the other hand, datasets with known sub-classes (letters-binary, letters-consonants, letters-vowels, yeast4, yeast6) tend to show good performance improvements when ICC is applied. Datasets with suspected sub-classes (al, al-location, prompting) also appear to be more likely to show performance improvements when ICC is applied.

To examine these theories more closely we also generate several synthetic datasets with certain known properties. We generate datasets with either 2 or 5 different classes using scikit-learn (Pedregosa et al. 2011) based on the algorithm of Guyon (Guyon 2003). Within these classes data is generated from either 1, 2, or 4 clusters. For the 2-class datasets the distribution of the class labels is set to 50/50, 80/20 or 95/5; for the 5-class datasets we set the distribution to 20/20/20/20/20, 10/10/20/20/40 or 5/5/20/30/40 in the five class datasets. Each dataset has six different real-valued attributes. The clusters are normally distributed and centered on the vertices of a plane. Each class is randomly assigned an equal number of clusters. While all of the details are not listed here, the biggest improvements in F1-score occur on the most imbalanced datasets. Furthermore, datasets with only 1 cluster per class tend to show less improvement than the other datasets.

Related Work

Intra-class clustering is related to several other techniques and domains. These can be divided into three general categories: sampling, cost-sensitive learning, and learning algorithm-specific methods. Sampling techniques include both undersampling the majority class(Liu, Wu, and Zhou 2009; Galar et al. 2012; Seiffert et al. 2010) and oversampling the minority class (Chawla et al. 2002).

SMOTE is one common technique which over-samples the minority class by artificially generating new minority samples. The samples are generated by finding the centroid of several nearest neighbors all of which belong to the minority class. SMOTE-boost extends SMOTE by adding a boosting algorithm which gives more emphasis to points that were mis-classified in an earlier iteration (Chawla et al. 2003). RUSBoost is one recent technique which randomly undersamples the majority class (Seiffert et al. 2010). RUSBoost also combines the under-sampling with a boosting algorithm to further improve performance. RaCOG applies undersampling to the majority class and oversampling the minority class while more closely adhering to the original distribution (Das, Krishnan, and Cook 2015).

All of these sampling techniques either discard potentially valuable information through under-sampling the majority class or generate potentially misleading information through oversampling the minority class. One of the benefits of intra-class clustering is that no information is discarded and no new instances are artificially generated.

Cost-sensitive learning also shares the same benefit of intra-class clustering in that the training set remains unchanged. One major drawback of cost-sensitive learning, however, is the need to determine the cost-matrix (Ling and Sheng 2011). Determining the true costs of mis-classifying an instance can be difficult, time-consuming and even impossible in some cases.

Learning algorithm-specific techniques also benefit, in many cases, from not changing the training set data (Joshi, Kumar, and Agarwal 2001; Wu and Chang 2003). However, they are not generalizable to other learning algorithms and are less applicable as a general class imbalance technique.

Decomposing classes through clustering has also been applied to improving the classification results of high bias classifiers like Naive Bayes (Vilalta, Achari, and Eick 2003). Their technique is similar to ours but is not applied to class imbalance. Instead it focuses solely on the benefits of adding sub-classes to high bias classifiers.

Conclusion

Skewed class distributions represent a significant challenge for traditional supervised machine learning algorithms. Although many techniques have been developed to address this problem, they all contain drawbacks that affect the training data distribution or prevent general use of the method. We have introduced a new set of techniques, intra-class clustering, which can be applied to any existing classifier and avoid these problems.

We hypothesized that ICC could improve classification performance through class balancing and that it could introduce decision boundaries that represent intrinsic sub-classes. Our experimental results support this hypothesis and show that ICC is effective on many different real-world datasets. Using synthetic data, we have explored different characteristic of datasets which may affect the performance of intra-class clustering. Preliminary results indicate that ICC performs best on datasets containing intrinsic sub-classes and on datasets with skewed class distributions. In

the future we plan to generate more synthetic datasets which we can use to draw statistically significant conclusions about which properties of a dataset affect the performance of ICC.

We have shown that using intra-class clustering to decompose the majority class(es) into smaller sub-classes prior to training a classifier does indeed lead to improved performance of the learned model. We have also shown that ICC can lead to a more balanced class distribution, find intrinsic sub-class boundaries, and improve performance by introducing more decision boundaries. Although all of the variations of the ICC algoritms can lead to improved performance we observed the ICC_EM_ALL technique to be the most consistent. If computational time is prohibitive, we recommend the ICC_FIX_ALL technique which is faster, possibly at the expense of stronger results.

References

Arthur, D., and Vassilvitskii, S. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–1035. Society for Industrial and Applied Mathematics.

Bhattacharyya, S.; Jha, S.; Tharakunnel, K.; and Westland, J. C. 2011. Data mining for credit card fraud: A comparative study. *Decision Support Systems* 50(3):602–613.

Caliński, T., and Harabasz, J. 1974. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods* 3(1):1–27.

Chang, C.-C., and Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2:27:1–27:27.

Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; and Kegelmeyer, W. P. 2002. Smote: synthetic minority over-sampling technique. *Journal of AJournal of Artificial Research* 16:321–357.

Chawla, N. V.; Lazarevic, A.; Hall, L. O.; and Bowyer, K. W. 2003. Smoteboost: Improving prediction of the minority class in boosting. In *European Conference on Principles of Data Mining and Knowledge Discovery*, 107–119. Springer.

Das, B.; Cook, D.; Krishnan, N.; and Schmitter-Edgecombe, M. 2016. One-class classification-based real-time activity error detection in smart homes. *IEEE Journal of Selected Topics in Signal Processing* 10(5):914–923.

Das, B.; Krishnan, N. C.; and Cook, D. J. 2015. Racog and wracog: Two probabilistic oversampling techniques. *IEEE Transactions on Knowledge and Data Engineering* 27(1):222–234.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research* 9:1871–1874.

Feuz, K. D., and Cook, D. J. 2015. Transfer learning across feature-rich heterogeneous feature spaces via feature-space remapping (fsr). *ACM Transactions on Intelligent Systems and Technology (TIST)* 6(1):3.

Feuz, K. D.; Cook, D. J.; Rosasco, C.; Robertson, K.; and Schmitter-Edgecombe, M. 2015. Automated detection of activity transitions for prompting. *IEEE Transactions on Human-Machine Systems* 45(5):575–585.

Galar, M.; Fernandez, A.; Barrenechea, E.; Bustince, H.; and Herrera, F. 2012. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42(4):463–484.

Guyon, I. 2003. Design of experiments of the nips 2003 variable selection benchmark. In *NIPS 2003 Workshop on Feature Extraction and Feature Selection*.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter* 11(1):10–18.

Joshi, M. V.; Kumar, V.; and Agarwal, R. C. 2001. Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, 257–264. IEEE.

Lichman, M. 2013. UCI machine learning repository.

Ling, C. X., and Sheng, V. S. 2011. Cost-sensitive learning. In *Encyclopedia of Machine Learning*. Springer. 231–235.

Liu, X. Y.; Wu, J.; and Zhou, Z. H. 2009. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39(2):539–550.

Mazurowski, M. A.; Habas, P. A.; Zurada, J. M.; Lo, J. Y.; Baker, J. A.; and Tourassi, G. D. 2008. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks* 21(2):427–436.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Seiffert, C.; Khoshgoftaar, T. M.; Hulse, J. V.; and Napolitano, A. 2010. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 40(1):185–197.

Stefanowski, J. 2013. Overlapping, rare examples and class decomposition in learning classifiers from imbalanced data. In *Emerging Paradigms in Machine Learning*. Springer. 277–306.

Vilalta, R.; Achari, M.-K.; and Eick, C. F. 2003. Class decomposition via clustering: a new framework for low-variance classifiers. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, 673–676. IEEE.

Wu, G., and Chang, E. Y. 2003. Class-boundary alignment for imbalanced dataset learning. In *ICML 2003 Workshop on Learning from Imbalanced Data Sets II, Washington, DC*, 49–56.