Nearest Neighbor Distributions for Imbalanced Classification

Evan Kriminger
and José C. Príncipe
Deparment of Electrical and Computer Engineering
University of Florida
Gainesville, Florida 32611–6200
Email: http://www.cnel.ufl.edu/

Choudur Lakshminarayan
HP Labs
Palo Alto, California
Email: Choudur.Lakshminarayan@hp.com

***Abstract*—The class imbalance problem is pervasive in machine learning. To accurately classify the minority class, current methods rely on sampling schemes to close the gap between classes, or on the application of error costs to create algorithms which favor the minority class. Since the sampling schemes and costs must be specified, these methods are highly dependent on the class distributions present in the training set. This makes them difficult to apply in settings where the level of imbalance changes, such as in online streaming data. Often they cannot handle multi-class problems. We present a novel single-class algorithm called Class Conditional Nearest Neighbor Distribution (CCNND), which mitigates the effects of class imbalance through local geometric structure in the data. Our algorithm can be applied seamlessly to problems with any level of imbalance or number of classes, and new examples are simply added to the training set. We show that it performs as well as or better than top sampling and cost-weighting methods on four imbalanced datasets from the UCI Machine Learning Repository, and then apply it to streaming data from the oil and gas industry alongside a modified nearest neighbor algorithm. Our algorithm's competitive performance relative to the state-of-the-art, coupled with its extremely simple implementation and automatic adjustment for minority classes, demonstrates that it is worth further study.**

I. Introduction

The performance of traditional classification methods is prone to deterioration when presented with significant class imbalance. Class imbalance occurs when the instances of one class are far less in number than the instances of another class. The class imbalance may refer to a relative imbalance, with class instance ratios on the order of 100 to 1, 1000 to 1, or even higher. However, it is often the case that the minority class has very few instances, so algorithms have to account both for relative imbalance and sparsity of instances in the minority class.

Class imbalance has attracted considerable attention in recent years. There have been various workshops, conferences, and special issues that have addressed the problem to summarize the vast array of emerging techniques, performance evaluation methods, and to point out important directions for research effort. This attention is well warranted as class imbalances are present in many real-world applications, including fraud/intrustion detection, anomaly detection, medical diagnosis [1]. In these fields, class imbalance is inherent, however imbalances may arise in any experiment where the class distribution is not explicitly controlled during data collection. Often it is the minority event which is so critical to detect. For instance, in a medical test, there will typically be significantly more "negative" than "positive" instances. Due to the scarcity of the positive class, a classifier which favors the negative class will produce an overall low error rate. However, false negatives are potentially catastrophic, while false positives simply warrant more testing. Thus it is clear that providing fair classification with respect to minority classes must be stressed.

Most of the methods designed to handle class imbalance fall into one of two categories: sampling methods and cost-sensitive methods [2]. Sampling methods operate on the data itself, attempting to reduce the imbalance between classes, by oversampling the minority class and/or undersampling the majority class. Cost-sensitive methods apply more weight to errors made on the minority class and may be applied to the data or incorporated into the classification algorithms themselves. For a review of the state-of-the-art in class imbalance methods, see [2].

Both sampling and cost-sensitive methods are tuned, either through the amount of sampling or through the relative costs assigned to each class, to provide the desired balance between classes. A third class of methods, which can be considered single-class classifiers consider whether or not to include an unlabeled sample into each class individually, rather than comparing to inter-class boundaries. Single-class methods are used extensively in anomaly detection, a field in which anomalies provide classes with few training examples [3]. Often these methods are useful in practice, since they easily tune to the degree of imbalance present in the training set. In many applications, the degree of imbalance will change, particularly when classifying online streaming data [2]. We propose the Class Conditional Nearest Neighbor Distribution (CCNND) algorithm, which naturally adjusts to any level of imbalance which may be present and boosts the performance with respect to the minority class. Furthermore, it is easily extended to the multiclass case.

This paper is arranged as follows: In Section II we describe our algorithm and provide some justification for its use. In Section III we demonstrate its performance on UCI datasets, which serve as benchmarks problems for binary, imbalanced classification, and compare to a few popular methods respresenting the techniques of sampling and cost-sensitivity. Finally, in Section IV we show how our method compares to the standard kNN classifier and a modified kNN classifier called the Neighbor-Weighted k-Nearest Neighbor (NWKNN) algorithm [4], in an online streaming data setting consisting of real-world, imbalanced, multi-class data from the oil industry.

II. Class Conditional Nearest Neighbor Distribution

A. Motivation

We present the Class Conditional Nearest Neighbor Distribution (CCNND) algorithm, which uses the structure of the data itself to discern and account for the imbalance in the data set. Class imbalance methods attempt to account for the disadvantage that the minority class faces in classification. Essentially, class imbalance methods push the classification boundary futher from the minority class, which increases the rate of true positives. We propose that an acceptable boundary need not be found through trial and error or expert knowledge. Rather we can extract a boundary which maintains high sensitivity to the positive class, directly from the data.

The boundary should be based on the variability that is to be expected when the class is populated by future examples. We represent the variability with the nearest neighbor distances. When a class has few representative samples, it is likely that these samples will be further apart than the samples of a more populated class, drawn from the same distribution. Likewise, when a class is well represented, the neighbor distances will be smaller. Therefore, the location of the classification boundary can be determined from the relative distance properties of each class.

Traditional methods, such as kNN, classify a sample by comparing its distances to each class directly. Nearest neighbor distances should not be compared directly because this places a boundary equidistant between samples of different classes, even if one class is much more likely to exceed this boundary in the future. In the presence of class imbalance, the majority class dominates a boundary that relies on equal weighting of distances. In order to compare nearest neighbor distances, we therefore consider them relative to the distances that are present amongst the instances of a given training set. This approach provides an empirical measure of how likely it is to see a new sample with certain nearest neighbor distances from a class, given the distribution of the existing distances for that class.

B. Method

Consider the training set $\mathcal{C} := \{C_i\}_1^L$, where C_i is class i, consisting of N_i data points $\mathbf{x} \in \mathbb{R}^m$. Let k be the number of neighbors which will be evaluated, as selected by the user. For each sample, let $\mathbf{d}_i(\mathbf{x})$ represent the vector consisting of the distances of $\mathbf{x}$ to its k nearest neighbors in class i (not including the distance of $\mathbf{x}$ to itself). For each class i, for all $\mathbf{x} \in C_i$, we calculate $\mathbf{d}_i(\mathbf{x})$, and build the empirical

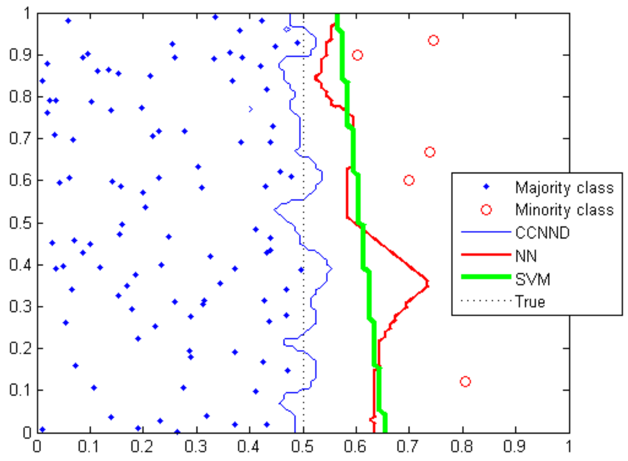

Fig. 1. Classification boundaries of CCNND with $k = 1$, 1-nearest neighbor, and a support vector machine classifier on a synthetic imbalanced dataset. The majority class consists of 100 samples uniformly distributed on the left half of the plane at $x = 0.5$. The minority class is made up of only 5 samples from a uniform distribution on the right side. The true boundary is along the dividing plane. The 1-NN and SVM classifiers place a boundary which favors the majority class. The CCNND boundary nearly approximates the true boundary.

CDF (cumulative distribution function) of nearest neighbor distances for class i.

When presented with an unlabeled testing sample, $\mathbf{x}_{\text{test}}$, for each class i, we calculate $\mathbf{d}_i(\mathbf{x}_{\text{test}})$. Using the empirical CDF of nearest neighbor distances for class i, we calculate the probability, $p_i(\mathbf{x}_{\text{test}})$, that a point with nearest neighbor distance $\mathbf{d}_i(\mathbf{x}_{\text{test}})$ would belong to that class, i.e. the probability that the nearest neighbor distances within a given class from training are greater than the nearest neighbor distances of the test point to the training set for that class.

For class i, consider the set of points in i, such that the nearest neighbor distances are pointwise-less than the nearest neighbor distances of the test point,

$$\mathcal{S}_i := \{\mathbf{x}_j \mid \mathbf{d}_i(\mathbf{x}_{\text{test}}) < \mathbf{d}_i(\mathbf{x}_j) \text{ for } \mathbf{x}_j \in C_i\}.$$

The probability, $p_i(\mathbf{x}_{\text{test}})$, may be computed

$$p_i(\mathbf{x}_{\text{test}}) = \frac{|\mathcal{S}_i|}{|C_i|},$$

where $|\cdot|$ is the set cardinality operator. The decision rule for assigning a label to $\mathbf{x}_{\text{test}}$ is

$$\text{label}_{\text{test}} = \underset{i \in [1 \ldots L]}{\arg\max}\; p_i(\mathbf{x}_{\text{test}}).$$

We would not be able to compare nearest neighbor distances in this way because of the explicit bias given to the majority class. Since we work with the probability of finding a nearest neighbor farther than the nearest neighbor of the test sample, this comparison is possible and provides adjustment to imbalance. Figure 1 provides a synthetic example demonstrating the

TABLE I

NUMBER OF POSITIVE AND NEGATIVE INSTANCES PRESENT IN EACH DATASET.

Dataset (pos. class)	Pos. Instances	Neg. Instances
Abalone (19)	32	4145
Glass (7)	29	185
Letter (26)	734	19266
Segment (1)	330	1980

ability of CCNND to handle extreme imbalance by creating a decision boundary based on the distance properties of the samples that have been observed, while kNN and SVM classifiers fail.

Selection of the number of neighbors, k, is similar as for kNN. From our experimental studies, small values of k (between 1 and 3), performed best due to the fact that the minority class has few samples to represent large areas of feature space.

III. COMPARISON WITH SDC

To evaluate CCNND we compare it with successful representatives of the sampling and cost-sensitive approaches to imbalanced classification. Synthetic Minority Over-sampling Technique (SMOTE) [5] is a popular sampling algorithm, which was combined with a cost-sensitive approach in the SMOTE with Different Costs (SDC) [6] algorithm. SDC offers the benefits of both sampling and cost-sensitive approaches and has been shown [6] to outperform undersampling, SMOTE, and Different Error Costs [7] which is a cost-sensitive method. We compare to SDC, SVM, and random undersampling, repeating those results of [6] for convenience.

SDC was tested on 10 datasets from the UCI Machine Learning Repository [8]. We focus on the 4 out of these 10 datasets with numerical attributes and no missing data because preprocessing methods to account for missing data and categorical variables were not specified. These are the *Abalone*, *Glass Identification*, *Letter Recognition*, and *Image Segmentation* datasets. These datasets represent various degrees of imbalance and dimensionality of the feature space. While these problems have multiple classes, in accordance with [6], we single out one of these classes as the "positive" class and combine the others together to form the "negative" class. In Table I, we show the degree of imbalance present in each dataset and the class which represents our minority class.

The ratio of training to testing data is 7 to 3, with these sets being selected randomly. However, the class ratios were kept the same in both the training and testing sets. The results were averaged over 100 trials. Each column of the feature vectors was normalized to 1 for the benefit of kNN. The value of k which produced the best results is shown, although no value showed substantially lower performance. The results are shown in Table II. Three measures of performance are used: sensitivity ($acc+$), which is the ratio of correct detections of the positive class to the total number of positive examples, specificity ($acc-$), which is the ratio of the correct detection of the negative class to the total number of negative examples, and the g-means metric [9], which is defined as

$$g = \sqrt{(acc+)(acc-)}.$$

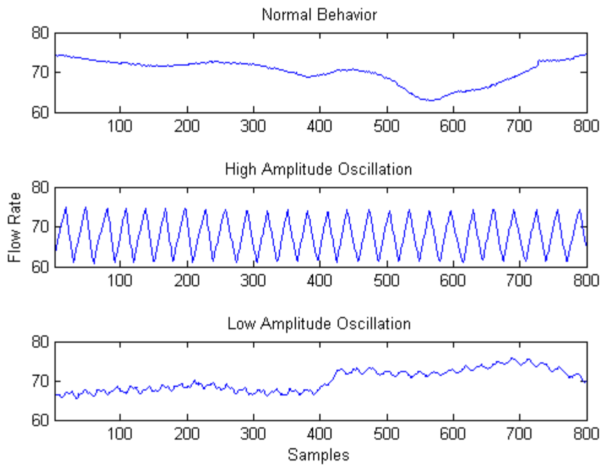

Fig. 2. Normal behavior (top) and the two oscillatory classes which indicate the onset of slugging and churn (middle and bottom).

The performance of CCNND is comparable to SDC and shows improvement in terms of g-means in 3 out of the 4 datasets. Both methods greatly outperform a standard SVM and random undersampling. There is a large increase in the sensitivity with CCNND on the Abalone and Glass datasets, demonstrating ability of CCNND to maintain a high true positive rate, while still limiting the number of false positives to within an acceptable range. Unlike SDC, which requires optimization of costs as well as oversampling of the minority classes, CCNND requires only evaluation of nearest neighbor distances.

IV. ONLINE STREAMING DATA APPLICATION: OIL FLOW-RATE

A. Data Description

We tested CCNND on a multi-class dataset derived from a real time series of oil flow rates from sensors placed in a borewell. In this example, the measurements are from a single well, recorded by a single sensor at 30 second intervals located at the surface of the sea. The detector must label each new data point that streams in as being in normal operation or to flag the onset of hazardous conditions, known as *slugging* and *churn*. The conditions for slugging and churn may be recognized in flow rate data by segments consisting of large amplitude, triangular oscillations, or of smaller oscillations superimposed on normal behavior. Example segments of normal behavior, high amplitude oscillation, and low amplitude oscillation are seen in Figure 2. It is important to quickly identify the two oscillatory behaviors that lead to slugging and churn so that appropriate actions may be taken to control it.

TABLE II
A PERFORMANCE COMPARISON CCNND, SDC, US, AND SVM ON FOUR UCI DATASETS. FOR THREE OF THESE DATASETS CCNND SCORES HIGHER THAN SDC IN TERMS OF THE G-MEANS METRIC AND SENSITIVITY. BOTH METHODS WELL OUTPERFORM US AND SVM.

	CCNND				SDC			US			SVM		
	Sens.	Spec.	g-means	k	Sens.	Spec.	g-means	Sens.	Spec.	g-means	Sens.	Spec.	g-means
Abal.	0.880	0.633	0.7462	2	0.808	0.687	0.7449	0.778	0.533	0.6436	0.889	0.732	0
Glas.	0.956	0.900	0.9274	2	0.808	1	0.9405	0.875	0.885	0.8801	0.75	1	0.8660
Lett.	0.978	0.999	0.9884	2	0.997	0.966	0.9817	0.996	0.917	0.9555	0.67	1	0.8183
Seg.	0.969	0.999	0.9835	1	0.959	0.998	0.9783	0.99	0.993	0.9918	0.99	1	0.9950

TABLE III
PERFORMANCE RESULTS OF CCNND, NWKNN, AND KNN ON THE OIL FLOW RATE DATASET. EACH TABLE IS A CONFUSION MATRIX, WITH THE ROWS REPRESENTING THE TRUE CLASS, AND THE COLUMNS REPRESENTING THE CLASS ASSIGNED BY EACH RESPECTIVE METHOD. WHILE NWKNN IMPROVES SIGNIFICANTLY ON KNN, CCNND PERFORMS BETTER ON THE MINORITY CLASSES AND IN TERMS OF OVERALL ERROR RATE.

(a) CCNND, $k = 1$, Error rate = 0.101

	Class 1	Class 2	Class 3
Class 1	0.789	0.073	0.138
Class 2	0.011	0.984	0.005
Class 3	0.076	0.001	0.923

(b) NWKNN, $k = 8$, $c = 2.5$, Error rate = 0.185

	Class 1	Class 2	Class 3
Class 1	0.950	0.011	0.039
Class 2	0.000	0.842	0.158
Class 3	0.326	0.120	0.654

(c) kNN, $k = 1$, Error rate = 0.370

	Class 1	Class 2	Class 3
Class 1	0.998	0.000	0.002
Class 2	0.002	0.727	0.371
Class 3	0.705	0.135	0.165

Classifiers that function in a streaming environment must be able to handle the appearance and disappearance of classes as conditions change and thus adapt to changing degrees of imbalance. Therefore, it is important that the classifier not depend on joint information between classes, so that the classifier can be updated by adding new data points, rather than completely retraining. The simple implementation and modular design of CCNND lends itself to streaming data applications.

B. Comparison to NWKNN and kNN

We compare CCNND to the Neighbor-Weighted k-Nearest Neighbor (NWKNN) algorithm of Tan [4], and to the standard kNN algorithm. NWKNN is a modification of kNN, in which a weight is computed for the distances of each class based on the ratio of the class population to the minority class. The weight for class i is,

$$w_i = \left(\frac{\min_{j=1,\ldots,L} N_j}{N_i} \right)^{\frac{1}{c}},$$

where $c > 1$ is a tunable parameter. If $\mathrm{kNN}_{\mathcal{C}}(\mathbf{x}_{\mathrm{test}})$ represents the k nearest neighbors of $\mathbf{x}_{\mathrm{test}}$, and we define

$$\mathcal{X}_i := \{\mathbf{x}\,|\mathbf{x} \in C_i \text{ and } \mathbf{x} \in \mathrm{kNN}_{\mathcal{C}}(\mathbf{x}_{\mathrm{test}})\},$$

then the score of class i is defined to be

$$score(i, \mathbf{x}_{\mathrm{test}}) = w_i \sum_{\mathbf{x} \in \mathcal{X}_i} \mathrm{sim}(\mathbf{x}_{\mathrm{test}}, \mathbf{x}),$$

where $\mathrm{sim}(\cdot)$ is a similarity measure. The test sample is given the label of the class with the highest score. By providing the weighting factor, the neighbors from minority classes contribute more to the score.

Each sample of the time series contributed a feature vectors consisting of a 6-dimensional vector consisting of the sample and the previous samples with time lags of 4 between them. Appended to the time embedding vector was a scalar representing the signal power in a window of 16 samples previous to the current sample. The training set consists of 1500 samples from normal behavior (Class 1), 50 from high amplitude oscillation (Class 2), and 10 from low amplitude oscillation (Class 3). The testing set consisted of 500 samples from each class. For CCNND and kNN, $k = 1$. For NWKNN, $k = 8$ and $c = 2.5$. The results, averaged over 100 trials, are seen in Table III.

V. CONCLUSION

We have presented the Class Conditional Nearest Neighbor Distribution (CCNND) algorithm, which accounts for class imbalances without the need to set tuning parameters. CCNND maintains high sensitivity to the minority class, while still providing high overall performance in comparison with popular methods. This feature is important in fields where false negatives can be damaging. CCNND is extremely simple to implement and to adapt to changes in imbalance. It is therefore very applicable in settings, such as in online streaming data. We have tested CCNND on UCI datasets and shown that the performance of CCNND is comparable to the state-of-the-art. The ability of CCNND to handle real-world, multi-class data was then demonstrated on oil pipeline data for the detection of dangerous slugging and churn behavior. In an imbalanced setting, CCNND greatly outperforms kNN, which fails to maintain high sensitivity to the minority class. CCNND also outperforms a similar single-class method, NWKNN, in this example. This paper has shown that the local geometric structure of a dataset provides insight into the variability that can be expected from future observations.

References

[1] N. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[2] H. He and E. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1263–1284, 2008.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, 2009.

[4] S. Tan, "Neighbor-weighted k-nearest neighbor for unbalanced text corpus," *Expert Syst. Appl.*, vol. 28, no. 4, pp. 667–671, 2005.

[5] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, 2002.

[6] R. Akbani, S. Kwek, and N. Japkowicz, "Applying support vector machines to imbalanced datasets," *Machine Learning: ECML 2004*, pp. 39–50, 2004.

[7] K. Veropoulos, C. Campbell, and N. Cristianini, "Controlling the sensitivity of support vector machines," in *Proceedings of the international joint conference on AI*. Citeseer, 1999.

[8] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[9] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection," in *Machine Learning - Int. Workshop then Conf.* Citeseer, 1997, pp. 179–186.