SMOTE: Synthetic Minority Over-sampling Technique

Siddhant Prakash, *1211092724* and Kunal Bansal, *1211XXXXXX*

Abstract—The abstract goes here.

Index Terms—Computer Society, IEEE, IEEEtran, journal, LaTeX, paper, template.

1 Introduction

THE seminal work, "SMOTE: synthetic minority oversampling technique" [1] deals with the fundamental problem of imbalanced class data-sets in machine learning, where, data points of one class are in majority over data points of another class. A number of times the data points belonging to the minority class are more important than the majority class. This majority-minority skew in the dataset leads to classification in favor of majority class samples when one uses standard classification techniques like Naive Bayes or Decision Trees. In several cases, the penalty for mis-classifying these minority classes are much higher than mis-classifying the majority class. For example, the problem of classifying the images of mammograms for cancerous cell detection is very sensitive and mis-classification may lead to disastrous outcomes. Still, the number of positive class samples is way outnumbered by the number of negative class samples with the majority (negative) class sample constituting 98% of total samples.

The authors in [1] propose a new algorithm to augment the minority samples in a data-set by creating synthetic data points for minority class to even the data distribution between majority and minority classes. They compare the result obtained by this new technique with the results of plain under-sampling the majority samples, as well as oversampling the minority samples as previously done in other works dealing with the problem. The results show that their method leads to learning of better classifiers. Further, they show that the classifiers improve with near equal representation from all class in training data. The authors use Receiver Operating Characteristics (ROC) curves as their performance measure. ROC curves provide trade-off between true positive (TP) vs. false positive (FP) which is much more suited to the class imbalance problem than the error rate (accuracy) metric. Area Under the Curve (AUC) and convex hull of ROC curve are used for the experiments as they provide good comparison of the classifiers' performance in class imbalance scenarios.

In our project, we learn to deal with the challenges of class imbalanced data-sets and how to overcome them. We reproduce the work done in [1] and understand the problem by going over the previous works cited in this paper. Additionally, we summarize the various techniques researchers came up with regards to dealing with this issue. We overview the performance metrics used in this paper and explain why the authors chose to use these metrics, as well as how are they relevant to this particular problem. We then move on to the implementation of the algorithm SMOTE [1] proposed in the paper, along with two more approaches which was used to emphasize the novelty of this technique. We replicate the experiments section on 3 of the 9 datasets listed in the paper, using Decision Trees, Nearest Neighbours and Nave Bayes classifiers to learn classification models. We also use the prediction results from these models to plot ROC curves, and taking AUC and convex hull of the curve as evaluation metric, compare our graphs with the graphs obtained in the paper.

2 Previous Works

Most of the cases of imbalanced dataset is dealt in two ways, viz. under-sampling the majority class samples or over-sampling the minority class samples. Different domain requires different techniques for the same based on their requirement. When it comes to under-sampling the majority class, Kubat et al (1997, 1998) experimented with the same. In Kubat and Matwin (1997), the majority class is selectively under sampled while the minority class sampling remains fixed. The performance metric used for the classifier is geometric mean which is not as expressive as a ROC curve and corresponds to just one point on it. Related to the above, the SHRINK system of Kubat et al (1998) classified the overlapping reasons of both majority and minority classes as positive, leading to a ”best positive region” classification.

Another study on under-sampling of dataset was performed by Japkowicz (2000). In her study, she explored different sampling techniques on artificial 1D data for better evaluation of concept complexity. Her exploration involved under-sampling as well as resampling of data. Both strategies involved two different methods, viz. random and focused. Random resampling used samples from minor class to be sampled randomly until they matched major class samples, while focused resampling used only the boundary

- *S. Prakash and K. Bansal are with School of Computing, Informatics and Design Systems Engineering, Arizona State University, Arizona, AZ, 85281.*
E-mail: {sprakas9, kbansal3}@asu.edu

points between minor and major class. In random under-sampling, the samples from majority class were removed randomly to match the minority class samples, in contrast to focused under-sampling which under sampled majority class samples lying further away. Her study revealed the efficacy of both the sampling techniques but did not provide any clear advantage in the domain considered.

While under sampling approach works, other works uses under-sampling of majority class samples along with over-sampling of minority class samples for learning a better classifier. Ling and Li et al (1998) uses lift analysis to measure classifier's performance in the domain of marketing analysis problem. They ranked the test examples by confidence measures and used lift as the evaluation criteria. In one of the experiments they performed, they under sampled the majority class and observed that the best lift index is obtained when there is equal representation of the classes. In another experiment, they over-sampled the minority samples with replacement to match the negative samples but could not prove the same as significant. The work present in this paper is similar in strategy, but the over-sampling techniques is different. Another work which uses the idea of under-sampling as well as over-sampling of data to overcome class imbalance problem is Solberg and Solberg et al (1996). They use SAR imagery dataset obtained for classification of oil slicks which is heavily biased towards look-alike data compared to oil slicks (98%-2%). They created a new dataset by over-sampling the oil slicks data randomly and under-sampling the look-alike data to create equal class distribution. As a result, on learning a classification tree on the balanced dataset they obtained better error rates on both classes compared to training on imbalanced dataset. Domingos et al (1999) also take the same approach to deal with class imbalance by introducing a "metacost" term to under-sampling as well as over-sampling. The work shows the metacost improves over either, and proves that under-sampling of majority class does better than over-sampling of minority class.

Other researchers (DeRouin et al 1991) tried to use the same on feed-forward neural networks which is not able to learn to discriminate between classes sufficiently due to the same class imbalance problem. The learning rate of the neural network was adapted according to the class distribution in the data set. Experimenting over artificial as well as real-world training data with multi-class problem provided better classification accuracy for minority class. In information retrieval domain, document classification is one of the challenging problems which is affected by this class imbalance. Creating a simple bag-of-words model results in interesting words samples as a minority due to very limited instances of such words in the document. Thus, in IR domain, the performance metric is replaced from error rates and instead, precision and recall terms are used for performance measurements.

$$recall = \frac{TP}{TP + FN}$$

$$precision = \frac{TP}{TP + FP}$$

We will see what we mean by the terms True Positive (TP), False Positive (FP) and False Negative (FN) in the next section. In the same domain, Mladenic and Grobelnik (1999) proposed a feature subset selection approach to deal with class imbalance. They found out that using odd ratio along with Naive Bayes classifier performs best in the domain. Odds ratio incorporates target class information giving better result over information gain which is computed per word for each class. In 1997, Provost and Fawcett introduced the ROC convex hull method for performance evaluation of the classifier in which ROC space is used to separate classification performance from class and cost distribution.

3 Evaluation Criteria and Performance Metrics

Confusion matrix, as shown in Table 1, is one of the most common method in machine learning used to evaluate performance of a (2-class) classification problem.

	Predicted Negative	Predicted Positive
Actual Negative	True Negative (TN)	False Positive (FP)
Actual Positive	False Negative (FN)	True Positive (TP)

TABLE 1
Confusion Matrix

As shown in the table, the columns are predicted class and the rows are actual class. Over a dataset of finite samples, the count of correctly classified negative samples is termed as True Negative (TN), while the count of incorrectly classified negative samples is termed as False Positive (FP). Similarly, the count of incorrectly classified positive sample is termed as False Negative (FN), while the count of correctly classified positive sample is termed as True Positive (TP).

For any classification task, predictive accuracy is defined as the total number of correctly classified samples over total number of samples. Mathematically, it is given by,

$$Accuracy = \frac{TP + TN}{TN + FP + FN + TP}$$

In machine learning, we evaluate the performance of a classifier by its error rate which is given by,

$$ErrorRate = 1 - Accuracy$$

This performance measure works well for balanced class data sets, but for imbalanced datasets a much wider used metric is the Receiver Operating Characteristics (ROC) curves.

A typically ROC curve is a plot of percentage True Positive vs percentage False Positive. One such curve is shown in Figure **XX**. We have %ge FP on the X-axis given by,

$$\%ge\ FP = \frac{FP}{TN + FP}$$

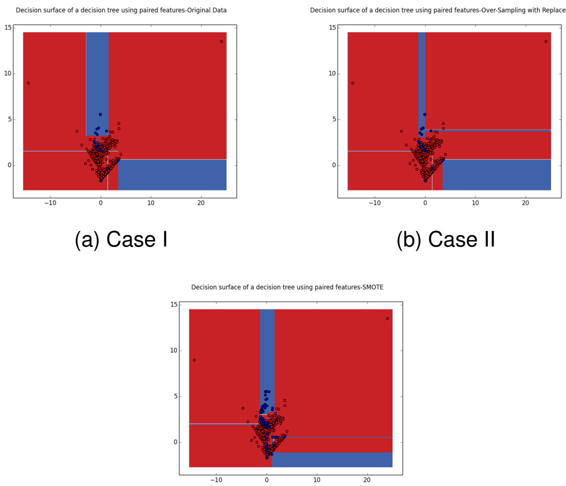
and %ge TP on the Y-axis given by,

$$\%ge\ TP = \frac{TP}{TP + FN}$$

As evident from the definition, the ideal point on the curve will be (0,100), signifying the all positive examples are classified correctly while no negative samples are classified wrongly as positive. The Area Under the Curve (AUC) can be taken as a good metric for comparing different classifiers, but these can be suboptimal for some specific cost and class distributions. Thus, the convex hull of ROC curve, being potentially optimal, is also taken as one of the performance metrics.

4 Implementation

In this section we provide the details of the various approaches we implemented from the paper. Using the "mammography" dataset [3], we show the effectiveness of SMOTE in this class imbalanced scenario. The dataset consist of 11,183 samples of which 10,923 are negative samples while we have only 260 positive samples. The dataset has 6 attributes, while we learn the classifier on 2 attributes, by decomposing the dataset using PCA, for better visualization. In Figure 1, we show the decision boundaries in the feature space, created by classifying 10% of the total data containing **18** positive samples and **1,100** negative samples. We have used the classifiers implemented in Scikit Learn library [2] with codes written in Python 2.7.1. The codes and dataset are provided in the supplementary material.



(a) Case I

(b) Case II

(c) Case III

Fig. 1. (a) Decision boundary on raw original data gives a biased classification towards negative (red) class samples.

4.1 Over-sampling with replacement

Learning a classifier on the imabalanced class data gives us a biased classification towards the majority class samples (red circles). From Figure 1a, we observe that the classification decision boundary using the original data is not very accurate. Many minority class samples (blue circles) are classified under majority class (red region) resulting in decrease in true positive rate. The decision boundary is more general and spread away from minority (positive) class samples too.

One of the earliest ways suggested by researchers to overcome this issue is over-sampling the minority class samples, in order to reduce the bias towards the majority class. We implement the over-sampling approach by duplicating the minority class samples. The degree of over-sampling in our implementation is 500%, i.e. if the dataset had 18 positive class samples and 1,100 negative class samples, as is the case in Figure 1, we upsample the positive class sample to 90. As a result, we see the decision boundary for minority class samples (blue region) shrink to enclose each individual positive sample. In other words, the data starts overfitting towards the positive class samples as more and more leaf nodes are added to the decision tree leading to very specific decision boundaries in the feature space. We see the same happening in Figure 1b with the 6 positive samples in the region around (0, 5).

4.2 SMOTE

Over-sampling of minority samples provides better accuracy but increases the complexity of classifier learnt manifolds. Moreover, the results does not generalize well as the decision region become very specific to the few minority samples present in the feature space leading to increase in false negatives. To overcome these issues a new over-sampling technique is proposed which augments the minority class samples by adding "synthetic" data. The algorithm works in the feature space of minority data samples. For a given degree of over-sampling rate, say 200%, the algorithm finds k-nearest neighbors of a minority sample and generates 2 random points on the line joining any 2 of the k-nearest neighbors found. We use nearest neighbors class implementation from scikit-learn library to find the nearest neighbor of each data point in the minority feature space. Algorithm 1 gives the steps to generate "SMOTED" data.

Figure 1c compares the effect of over-sampled data using SMOTE by 500%, with original data (Figure 1a) and replaced data (Figure 1b). It is clear from the plot that synthetic data obtained using SMOTE helps classifier to generalize well over minority samples by providing better synthetic data to overcome the class imbalance. The decision region becomes larger and less specific as is the case with over-sampling by replacement, at the same time reducing mis-classification by creating better decision boundary which is seen in the case of classification using original data.

4.3 SMOTE with under-sampling

Over-sampling minority class samples with the help of SMOTE helps in increasing the percentage of minority samples, but still doesn't help overcoming the effect of grossly skewed data perfectly. To overcome this bias towards majority class sample, we under-sample the majority samples in addition to SMOTing the dataset. The under-sampling is done to a degree with respect to the number of minority class samples. We first SMOTE the dataset and obtain the final number of minority class samples and then under-sample the majority class sample to get the desired degree of under-sampling. For example, in the above example, we have 18 minority samples and 1,100 majority samples. SMOTing the dataset by 500% results in an increase in minority samples to 90. Now, under-sampling the dataset by a degree of 200% means drawing out *half the samples of minority class samples, at random, from the total majority class samples*, i.e using 45 samples from the 1,100 majority (negative) class samples along with the 90 minority (positive) class samples to draw the decision region.

```
Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while not at end of this document do
    read current;
    if understand then
        go to next section;
        current section becomes this one;
    else
        go back to the beginning of current section;
    end
end
```

Algorithm 1: Over-sampling using SMOTE - Algorithm

5 Experiments

We experiment with our implementation of plain under-sampling and under-sampling with SMOTE under varying degrees of under-sampling and SMOTE combinations. Our aim is to tackle the class imbalance problems in datasets, thus, we test the algorithm on 3 different dataset with different class distributions. We use Decision Trees, Naive Bayes and k-NN classifiers with varying under-sampling and SMOTE degrees to (i) analyze the effectiveness of SMOTing data, and (ii) compare the classifiers learning the best decision boundaries. We plot the ROC curves and use ROC Convex Hull (ROCCH) and Area Under the Curve (AUC) as our metrics to draw our conclusions.

5.1 Datasets

The 3 dataset we use are Mammography Dataset [3], Pima Indian dataset [4] and SatImage dataset [5]. The class distribution of the datasets are provided in Table 2. We have provided the codes to convert raw data file into comma separated file format (CSV), which is the input dataset file format in our implementation and experiments code, in the supplementary material.

Each datasets are multi-dimensional, with mammography dataset consisting 6 attributes, Pima Indian having 8 attributes and SatImage datasets having 36 dimensions. Wherever needed, we reduce the dimensionality of dataset using PCA for preserving maximum information, faster computation and visualization of decision boundary in 2-D feature space.

5.2 Classifiers

Our main classifier for testing the algorithm implementation is **Decision Trees**. In the paper, the authors use C4.5 variant of Decision Trees to test the classification quality. We have used the implementation of Decision Trees provided by scikit-learn library, which is an optimized version of C4.5 algorithm called Classification and Regression Trees (CART) algorithm. As stated in the documentation [6], "CART is very similar to C4.5, but it differs in that it supports numerical target and does not compute rule sets. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node."

Datasets	#Minority (+ve) Class	#Majority (-ve) Class
Mammography	260	10,923
SatImage	626	5,809
Pima Indian	268	500

TABLE 2
Class Distribution in Datasets Used

In addition to decision tree classifiers, we compare the performance of the same algorithms on **Naive Bayes** classifier with varying minority class priors. The paper also uses Ripper classifier to compare the performance, but we failed to locate any existing implementation of Ripper classifier in a 3rd party library in Python. Instead we use **k-Nearest Neighbor** classifier with a fixed value of k for comparison with Decision Tree classifier, and varying values of k for comparison amongst k-NN classifiers. Our change in classifier is guided by the intuition that the algorithm augments data in the feature space, and since k-NN classification works directly in the feature space, it should provide a better classification accuracy.

5.3 ROC Curves and Cross-Validation

In order to evaluate the performance we plot the Receiver Operating Characteristics (ROC) curves for varying parameters of classification. ROC curves are essentially a plot of False Positive Rate (FPR) on x-axis vs True Positive Rate (TPR) on y-axis which measures a classifier's performance in more detail than a simple accuracy measure. This granularity is of great use in a class imbalance scenario because it gives an idea about the degree of mis-classification in both positive as well as negative class. Ideally, we will like a low False Positive Rate and a high True Positive Rate, resulting in a shift in ROC curve towards the $(0, 100)$ point. The nearer the curve to the same point, the better the classification.

In our experiments we draw two plots. The first plot contains Decision Tree as the base classifier and we compare the Decision Tree implementation of plain under-sampling of majority class samples at rates varying from **10%** to **2000%** as described in Section 4.3. For plain under-sampling we do not SMOTE the dataset before under-sampling. We compare the ROC curve obtained by plain under-sampling with that obtained by under-sampling with SMOTE. To obtain the second ROC curve, we over-sample the dataset with varying degree from **100%** to **500%**, based on the dataset class distributions. In the same plot we compare Decision Tree classifier with Naive Bayes classifier, by adding another ROC curve plotted using classification from Naive Bayes with varying class priors of minority class. In the implementation provided by scikit-learn, the class priors are automatically calculated as the frequency of each class samples, as stated in the documentation [7]. Thus, we feed in the data used in plain under-sampling to NB classifier as it varies the ratio between the two class samples. The various class priors used to plot the curve vary for different dataset based on the class distribution ratio of the dataset. We report the class priors at the end, after performing each experiment. In the second plot we replace decision tree classifier with k-NN classifier as the base classifier. We provide a comparison of the classifier with respect to plain under-sampled data and under-sampled data with SMOTE as with previous case.

Each point on the ROC curve is obtained by classifying the data using a **10-fold cross validation**. In an n-fold cross validation, each dataset is divided into n equal parts randomly, and (n-1) of those parts are use for training while 1 left-out part is used for testing. This process is iteratively performed n times and classification metric is generated with each prediction. The points are the mean False Positive Rate vs mean True Positive Rate for the 10 different classification with a particular configuration of over-sampling/ under-sampling combination. Figure **xx, yy and zz** provides the ROC curves we plot with a comparison on all 3 dataset and the 3 classifiers we use.

5.4 Convex Hull and AUC

The performance of the ROC curve is quantified using ROC Convex Hull and AUC metric. Convex Hull gives the boundary of a potential optimal classifier in comparison of multiple ROC curves. ROC Curves themselves may not be optimal at every instance. Thus, we use the hull to find out when a given classifier/method is optimal and when it can be replaced. We have implemented convex hull in all our plots, which should be taken as the perfect classification ROC curve. We construct the hull by first creating an array of all the 2D points on the ROC Curves using all classifier/method combination. Then we call the ConvexHull implementation of Scipy library [8] provided in *scipy.spatial* class, which essentially uses the QuickHull algorithm of [9]. This breaks the decision space into multiple simplices of the convex hull and finally we draw each individual simplex to complete the convex hull of the curves.

Area Under-the-Curve (AUC) is another metric which we use to evaluate the performance of each classification. In simple terms, the more the value of AUC, the better the classification as the curve will be nearer to the ideal point (0, 100) to maximize the area. We again use the implementation provided in scikit-learn library to calculate the AUC for each ROC curve. We summarize the AUC values in Table **??**.

6 Discussions

7 Conclusion

The conclusion and future scope goes here.

Acknowledgments

The authors would like to thank . . .

References

[1] Chawla, Nitesh V., et al. ”SMOTE: synthetic minority oversampling technique.” Journal of artificial intelligence research 16 (2002): 321-357.
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]