

# Generating Diverse Ensembles to Counter the Problem of Class Imbalance

T. Ryan Hoens<sup>1</sup> and Nitesh V. Chawla<sup>1</sup>

The University of Notre Dame, Notre Dame, IN 46556, USA  
{thoens, nchawla}@nd.edu

**Abstract.** One of the more challenging problems faced by the data mining community is that of imbalanced datasets. In imbalanced datasets one class (sometimes severely) outnumbers the other class, causing correct, and useful predictions to be difficult to achieve. In order to combat this, many techniques have been proposed, especially centered around sampling methods. In this paper we propose an ensemble framework that combines random subspaces with sampling to overcome the class imbalance problem. We then experimentally verify this technique on a wide variety of datasets. We conclude by analyzing the performance of the ensembles, and showing that, overall, our technique provides a significant improvement.

## 1 Introduction

A common problem faced in data mining is dealing with “imbalanced datasets”, or datasets in which one class vastly outnumbers the other in the training data. For most classifiers this causes a problem, as merely classifying every instance as the majority class present in the training data can result in very high accuracy. Therefore when dealing with imbalanced datasets, sampling methods such as oversampling, undersampling, and sampling by synthetically generating instances, SMOTE, have become standard approaches for improving classification performance [1]. Performance is improved as the sampling methods alter the training data distribution, biasing the data towards the minority class.

In addition to sampling methods, ensemble methods [2] have also been used to improve performance on imbalanced datasets. They combine the power of multiple (usually weak) classifiers trained on similar datasets to provide accurate predictions for future instances. The training data is often varied in such a way as to give each classifier a (slightly) different dataset so as to avoid overfitting. The popular ensemble methods, bagging [3] and boosting [4], have been adapted with sampling strategies to counter the issue of high class imbalance [5–7]. These methods work on the entire feature space as they focus on modifying the selection of instances in the ensemble training sets.

We posit that sampling methods, especially SMOTE, might stand to gain more when working in a reduced feature space as doing so will not only inject more diversity into the ensemble via the learning algorithm, but also via the bias of the sampling algorithm. To this end, we propose an extension to the random subspace method [8] that integrates SMOTE (in Section 2). We test on 21 widely available datasets with varying degrees

of class imbalance (Section 4 includes the results). We comprehensively compare the proposed random subspace and SMOTE combination with bagging, AdaBoost, random subspaces, random forest, and the combination of random subspace and undersampling. Using the statistical tests recommended by Demšar, we determine the statistical significance of results. We show that sampling methods, combined with random subspaces are more effective than the other combinations of ensemble and sampling methods. We explain this behavior by invoking the notion of diversity.

To summarize, **the contributions** of this paper are as follows:

1. An extension to the random subspace method to deal with the class imbalance problem.
2. Empirical evaluation on a wide variety of imbalanced datasets, and establishing the superiority of the new ensemble framework.
3. Analyzing the performance of the methods using the measures of diversity.

## 2 Using Random Subspaces to Improve Performance on Imbalanced Datasets

The random subspace method (RSM) [8], described in Algorithm 1, is an ensemble method in which multiple classifiers are learned on the same dataset, but each classifier only actively uses a subset of the available features.

Because each classifier learns on incomplete data, each individual classifier is less effective than a single classifier trained on all of the data. However since the RSM combines multiple classifiers of this type, each with its own bias based on the features it sees, it sees an increase in performance over the base classifier.

---

**Algorithm 1** The random subspace method.

---

**Require:** Training set  $X$  with  $n$  instances and  $m$  features, number of features to consider  $0 < m' < m$ , and number of classifiers to train  $e > 0$ .

**Ensure:** CLASSIFIER is the model trained on training set  $X$ , consisting of  $e$  classifiers.

```

for  $i = 1$  to  $e$  do
  Select  $F$ , a random subset of the features such that  $|F| = m'$ .
  Let  $X' \leftarrow X$ .
  for all  $f$  such that  $f$  is a feature of  $X'$  do
    if  $f \notin F$  then
      Remove feature  $f$  from  $X'$ 
    end if
  end for
  Train CLASSIFIER $_i$  on dataset  $X'$ .
end for

```

---

### 2.1 Random Subspace Method + Sampling

Sampling is a popular methodology to counter the problem of class imbalance. However sampling methods, especially SMOTE, work with the entire set of features or dimensions and may not be as effective in reducing the sparsity of the data. High

---

**Algorithm 2** The random subspace method with an added resampling step.

---

**Require:** Training set  $X$  with  $n$  instances and  $m$  features, number of features to consider  $0 < m' < m$ , number of classifiers to train  $e > 0$ , and sampling function  $S$  which takes a dataset as a parameter, and returns a dataset.

**Ensure:** CLASSIFIER is the model trained on training set  $X$ , consisting of  $e$  classifiers.

```
for  $i = 1$  to  $e$  do
  Select  $F$ , a random subset of the features such that  $|F| = m'$ .
  Let  $X' \leftarrow X$ .
  for all  $f$  such that  $f$  is a feature of  $X'$  do
    if  $f \notin F$  then
      Remove feature  $f$  from  $X'$ 
    end if
  end for
   $X'_s \leftarrow S(X')$ .
  Train CLASSIFIER $_i$  on dataset  $X'_s$ .
end for
```

---

dimensionality is an Achilles' heel for sampling approaches like SMOTE, as they can lead to a higher degree of variance given low density and separation among classes because of the features' spread. Intuitively, applying SMOTE to reduced subspaces at a time will also control the amount of variance that SMOTE may introduce.

Sampling methods consider the class skew and properties of the dataset as a whole. Datasets often exhibit characteristics and properties at a local, rather than global level. Hence, it becomes important to analyze and consider the datasets in the reduced subspace. We also posit that by first randomly selecting a reduced subspace, sampling along that subspace, and then learning a classifier will induce a higher diversity, which is a necessary condition for improved performance of classifiers.

To this end we propose using SMOTE within each randomly selected subspace. Since SMOTE creates synthetic instances by interpolating feature values based on neighbors, we see that it is dependent upon the distance metric used to determine nearest neighbors. Thus by removing features from the feature space, we see that we are altering the distance between instances, and therefore (potentially) changing the way in which each classifier modifies its training data. That is, we create a hybrid of the RSM by combining it with SMOTE to create RSM+SMOTE. Just like in the RSM during the training phase each classifier is trained on a subset of the data in which some features are removed. After removing a subset of the features, SMOTE is then applied to the dataset which is subsequently used to train the classifier. Since SMOTE is dependent upon the features, and since in ensemble methods having classifiers with different biases is optimal, this should provide better performance over other techniques.

A generalized version of this algorithm (RSM+sampling) is given in Algorithm 2. In this algorithm as opposed to explicitly using SMOTE, a generic sampling method is supplied as a parameter for use in the algorithm. That is, RSM+SMOTE is a special case of RSM+sampling carried with SMOTE as the sampling method. In our tests we also consider using random undersampling as the sampling method. This lacks the appeal of SMOTE, however, as it does not depend upon the subspace chosen, and is thus more like random forests.

Note that while in principal any classifier can be learned, we use C4.5 decision trees as is common in the literature. In order to determine the predicted class of the ensemble, we consider simple majority voting of the classifiers, similar to the other ensemble methods discussed. As an area of future work Chawla and Sylvester [9] outline a procedure for weighted voting as applied to imbalance datasets. We will also be including Hellinger Distance Decision Trees as part of future work [10].

**Complexity Analysis** When discussing ensemble methods, an important consideration is the method’s computational complexity. In this section we therefore give an overview of the complexity of the scheme considered in the paper. That is we only consider the case of RSM+SMOTE with C4.5 decision trees as the classifier.

Let us now consider building an ensemble of  $e$  C4.5 decision trees using random subspaces and SMOTE on a dataset  $D$  consisting of  $n$  instances and  $m$  features. From the pseudo-code given in Algorithm 2, we see that each iteration of the for-loop first selects  $m'$  features  $O(m')$ . The algorithm then (in our case) applies SMOTE which has a complexity of  $O(n_p^2)$ , where  $n_p$  denotes the number of positive (minority) class examples in  $D$ . Applying SMOTE to the dataset results in additional instances being added to the dataset, leaving us with  $n'$  instances. Finally a C4.5 decision tree is learned on the resulting dataset which has complexity  $O(n'm' \log n')$ . Combining this, we see that each iteration of the loop has a complexity of  $O(m' + n_p^2 + n'm' \log n') = O(n_p^2 + n'm' \log n')$ . Therefore the complexity of constructing RSM+SMOTE is  $O(e \cdot (n_p^2 + n'm' \log n'))$ .

As with most ensembles, we can achieve an order  $e$  speedup to our method if we instead build each tree in parallel. That is we break the task up into  $e$  jobs and distribute the tree building process to  $e$  cores. Additionally since the voting function is simple majority voting, the trees can remain distributed if the cost of transferring them is too high.

Full Name	Abbreviation
Bagging	BG
AdaBoost	BT
Random Subspace Method	RSM
Random Forest	RF
Synthetic Minority Over-sampling Technique	SMOTE
Undersampling	usamp
Random Subspace Method with SMOTE	RSM+SMOTE
Random Subspace Method with Undersampling	RSM+usamp

**Table 1.** Legend of abbreviations.

Dataset	# Features	# Examples	CV	Dataset	# Features	# Examples	CV
boundary	175	3505	0.93	hypo	25	3163	0.90
breast-w	9	699	0.31	ism	6	11180	0.95
breast-y	9	286	0.41	oil	49	937	0.91
cam	132	18916	0.90	page	10	5473	0.80
compustat	20	13657	0.92	phoneme	5	5404	0.41
covtype	10	38500	0.86	PhosS	480	11411	0.89
credit-g	20	1000	0.40	pima	8	768	0.30
estate	12	5322	0.76	satimage	36	6430	0.81
fourclass	2	862	0.29	SVMguide1	4	3089	0.29
germannumer	24	1000	0.40	tic-tac-toe	9	958	0.31
heart-v	13	200	0.49				

**Table 2.** Details of the datasets in this report.

### 3 Experimental Design

We used the open source tool Weka, and implemented our classifier RSM+sampling. In order to test the robustness of our method compared to existing methods, we included AdaBoostM1 (BT), bagging (BG), Random Forests (RF), and Random Subspaces (RSM) in our experiments. We also applied SMOTE and undersampling (us-amp) to the entire dataset and then learned a single classifier. For all of the methods we use the J48 decision tree as the base classifier with Laplace smoothing and no pruning. As each of the ensemble methods requires a number of classifiers to train, we train 100 classifiers for each. See Table 1 for the entire set of classifiers used in this paper.

We evaluated each of the classifiers on the twenty-one datasets from a number of different resources, including finance, biology, oil spill, medicine, UCI and LibSVM data repositories (Table 2) [10–12]. Similar to [13], we use the coefficient of variation (“CV”) to measure imbalance. To determine the CV, we calculate the ratio of the mean and standard deviation of the class counts in the dataset. As we seek to determine our classifiers’ performance on imbalanced datasets, we choose to test on datasets with a CV above 0.28. This corresponds to a dataset for which less than 35% of the instances are minority class. In general, the larger the value of CV, the more imbalanced the dataset.

#### 3.1 Experiments

In order to compare the classifiers, we use 10-fold cross validation. In 10-fold cross validation, each dataset is broken into 10 disjoint sets such that each set has (roughly) the same distribution. The classifier is learned 10 times such that in each iteration a different set is withheld from the training phase, and used instead to test the classifier. We then compute the AUROC (Area Under the Receiver Operating Characteristic) value as the average of each of these runs.

### 3.2 Statistical Tests

While many different techniques have been applied to attempt to compare classifier performance across multiple datasets, Demšar suggests comparisons based on ranks. Following the strategy outlined in [14], we rank the performance of each classifier by its average AUROC, with 1 being the best. Since we seek to determine whether or not our new methods are statistically significantly better than the existing methods, we use the Friedman and Bonferroni-Dunn tests [14].

The Friedman test is first applied to determine if there is a statistically significant difference between the rankings of the classifiers. That is, it tests to see if the rankings are not merely randomly distributed. Next, as recommended by Demšar, we perform the Bonferroni-Dunn test to compare each classifier against the control classifier.

## 4 Results

	RSM+SMOTE	RSM+usamp	RF	RSM	BG	BT
breast-w	2.0	1.0	3.0	4.0	5.0	6.0
breast-y	3.0	1.0	5.0	2.0	4.0	6.0
pima	1.0	2.0	5.0	3.0	4.0	6.0
hypo	1.0	2.0	5.0	3.0	4.0	6.0
phoneme	3.0	5.0	1.0	4.0	2.0	6.0
ism	1.0	2.0	4.0	3.0	5.0	6.0
SVMguide1	2.0	4.0	5.0	3.0	1.0	6.0
cam	1.0	2.0	5.0	3.0	4.0	6.0
credit-g	1.0	3.0	4.0	2.0	5.0	6.0
ion	1.0	3.0	4.0	2.0	5.0	6.0
covtype	1.0	5.0	2.0	4.0	3.0	6.0
satimage	2.0	3.0	1.0	4.0	5.0	6.0
PhosS	2.0	1.0	5.0	3.0	4.0	6.0
fourclass	4.0	5.0	1.0	6.0	3.0	2.0
tic-tac-toe	3.0	5.0	1.0	6.0	2.0	4.0
compustat	1.0	4.0	2.0	5.0	3.0	6.0
estate	1.0	3.0	4.0	5.0	2.0	6.0
heart-v	2.0	1.0	5.0	3.0	4.0	6.0
boundary	1.0	2.0	4.0	3.0	5.0	6.0
oil	1.0	3.0	4.0	2.0	5.0	6.0
page	1.0	2.5	5.0	2.5	4.0	6.0
Average	1.667	2.833	3.571	3.452	3.762	5.714

**Table 3.** Rank of each classifier on the datasets.

From Table 3 it is apparent that RSM+SMOTE performs significantly better than the other classifiers, achieving an average ranking that is over a full rank better than the next best classifier, RSM+usamp. This is further reinforced by noting that RSM+SMOTE is the best classifier in 12 of the 21 datasets, and second best on 5. RSM+SMOTE not only outperforms bagging and boosting, but also applying SMOTE on the entire dataset

before learning a classifier. The results of using SMOTE on the entire dataset are not included in this paper due to space restrictions, as well as to keep the focus on ensemble based methods.

Out of all of the datasets, RSM+SMOTE achieves its worst rank of 4 on the four-class dataset. This is unsurprising, however, as fourclass is not a suitable dataset to use RSM+SMOTE on, as it only has 2 features. Since RSM chooses a subspace of the features to learn on, it is necessary that there are enough features to gain power from only considering such a subset. Since fourclass only has 2 features, however, each classifier only classifies based on one feature. Similarly when SMOTE is applied to the feature there is only one axis of freedom, and the effect is that of placing instances somewhat randomly on a line. Intuitively this does not seem like an optimal strategy, and explains the poor performance of the classifier on the dataset. This also leads us to not recommend using RSM+SMOTE on such low dimensional datasets.

RSM+usamp also performs very well when compared to the other classifiers. On 10 of the 21 datasets it obtains the best or second best AUROC among the tested methods. This points to the sampling of individual subspaces to be a robust technique, as the two presented techniques offer a great advantage over the other classifiers.

Confidence Level	RSM+usamp	RF	RSM	BG	BT
99%		✓	✓	✓	✓
95%	✓	✓	✓	✓	✓
90%	✓	✓	✓	✓	✓

**Table 4.** Table denoting whether or not RSM+SMOTE is statistically significantly better than all other classifiers at various confidence levels on the datasets.

In order to measure the statistical significance of the results, we use the methods outlined in Section 3.2. The results of the tests are presented in Table 4. In this figure we show the results of the Friedman and Bonferroni-Dunn tests which show that RSM+SMOTE performs statistically significantly better than all classifiers at the 95% confidence level. At 99% confidence it outperforms all of the classifiers except for RSM+usamp. This is a strong result as it shows that the RSM can benefit greatly from applying sampling at the individual classifier level in addition to using a subspace of the features.

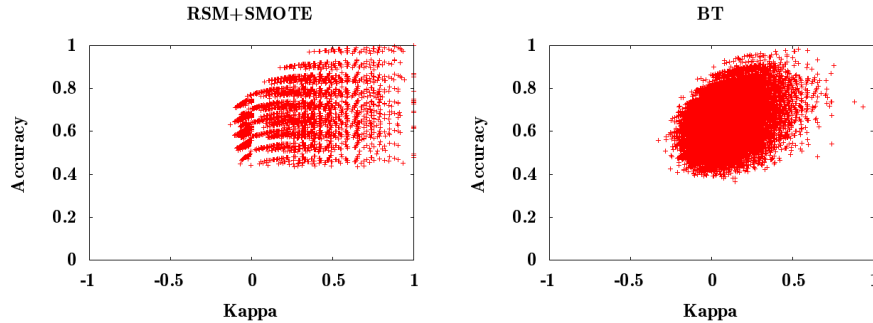
## 5 Analysis of Ensembles Using Diversity

When considering an ensemble of classifiers, the ideal situation would be a case where no two classifiers agree on instances on which they err. Given the nature of classifiers and data, however, achieving this ideal is highly unlikely. In order to determine how close an ensemble comes to the ideal, we measure the diversity by the degree to which an ensemble of classifiers disagree on such instances. Kuncheva and Whitaker [15] show that such diversity is an important property for achieving a good performance from ensembles.

In order to measure the diversity of the ensembles, we use the  $\kappa$  metric defined by Dietterich [16] as:

$$\begin{aligned}\theta_1 &= \frac{\sum_{i=1}^T C_{ii}}{m}, \\ \theta_2 &= \sum_{i=1}^T \left( \frac{\sum_{j=1}^T C_{ij}}{m} \frac{\sum_{j=1}^T C_{ji}}{m} \right), \\ \kappa &= \frac{\theta_1 - \theta_2}{1 - \theta_2},\end{aligned}$$

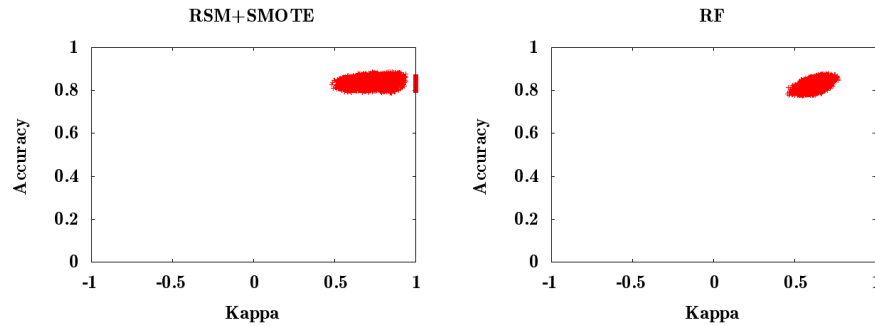
where  $T$  is the number of classes,  $C$  is a  $T \times T$  matrix such that  $C_{ij}$  denotes the number of instances assigned to class  $i$  by the first classifier and class  $j$  by the second classifier, and  $m$  is the number of instances. Given this,  $\theta_1$  measures the degree of agreement, and  $\theta_2$  is the degree of agreement expected at random.  $\kappa$  is then the measure of diversity, where 0 denotes agreement purely by chance, and 1 (−1) denotes perfect agreement (disagreement). The  $\kappa$  values can then be plotted against the accuracy for each pair of classifiers in the ensemble in order to obtain a graphical representation of the diversity of the ensemble. The x-axis is the  $\kappa$  value and the y-axis is the accuracy.



**Fig. 1.**  $\kappa$  plots for RSM+SMOTE (left) and AdaBoost (right) on the oil dataset.

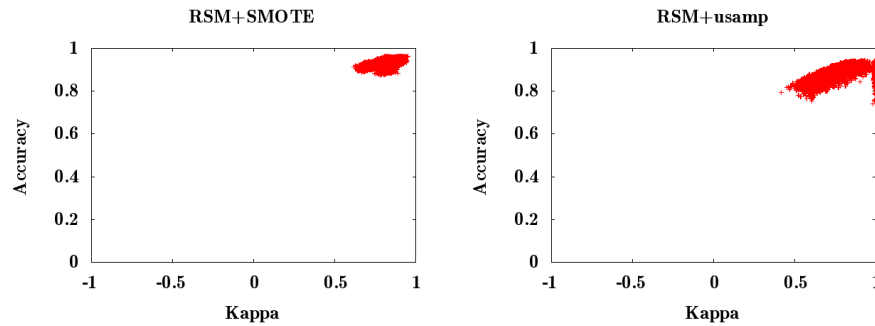
We now show some of the representative results explaining the behavior. We consider three representative results. Figure 1 shows the  $\kappa$  plot of RSM+SMOTE and AdaBoost on the oil dataset. Given the two plots, it becomes apparent why RSM+SMOTE outperforms AdaBoost on this dataset. While RSM+SMOTE shows some agreement among the classifiers along with high accuracy, AdaBoost shows much lower agreement (in fact almost completely random agreement) with similar accuracy. Since the classifiers in AdaBoost are agreeing almost at random, and the problem is very imbalanced, this is close to what we would expect of random classifiers which are biased towards the majority class. Alternatively, RSM+SMOTE shows very high accuracy with non negligible agreement. This points to highly accurate classifiers which agree when they are correct, and disagree on incorrectly classified instances, as desired.

While the previous example showed RSM+SMOTE performing well, Figure 2 depicts it being outperformed by RF. While AdaBoost underperformed for being too dissimilar, RSM+SMOTE underperformed for being too similar. As can be seen in the figure, RSM+SMOTE has a cluster of points at  $\kappa = 1$ . This means that the classifiers



**Fig. 2.**  $\kappa$  plots for RSM+SMOTE (left) and RF (right) on the phoneme dataset.

were in perfect agreement on every example. This adversely affects the diversity of the ensemble, effectively adding weighted classifiers to the ensemble. RF, on the other hand, shows highly accurate, yet diverse, classifiers.



**Fig. 3.**  $\kappa$  plots for RSM+SMOTE (left) and RSM+usamp (right) on the covtype dataset.

Finally we show an instance where RSM+SMOTE outperforms RSM+usamp in Figure 3. This plot shows precisely the desired results for RSM+SMOTE. That is, since the classifiers are highly accurate, they should have a high  $\kappa$  value which is approximately twice the error rate. While the  $\kappa$  values are in the higher strata, none of them are actually 1, which means that classifiers never have a perfect agreement and are diverse, despite high accuracies. RSM+usamp, on the other hand, shows similar high accuracies, yet many classifiers overlap as demonstrated by the number of points at  $\kappa = 1$ . As aforementioned this directly affects the diversity, and it is therefore unsurprising that RSM+SMOTE outperformed RSM+usamp.

## 6 Related Work

One popular approach towards improving performance on classification problems is to use ensembles. When using ensembles one attempts to leverage the classification power of multiple classifiers (learned on different subsets of the training data), to overcome the downsides of traditional classification algorithms, such as over fitting. Dietterich

[2] provides a broad overview as to why ensemble methods often outperform a single classifier. In fact Hansen and Salamon [17] prove that under certain constraints (the average error rate is less than 50% and each classifier is erroneous independent of the others), the expected error rate of an instance can go to zero as the number of classifiers goes to infinity. Thus when seeking to build multiple classifiers, it is optimal to ensure that the classifiers are diverse. One way of ensuring this is by modifying the training data each classifier is learned on.

Two techniques for modifying the training data used by each classifier are bagging [3] and AdaBoost [4]. In bagging, introduced by Breiman, each training set is chosen by sampling (with replacement) from the original training set  $T$ . In AdaBoost introduced by Freund and Schapire, however, each training set is iteratively chosen based on the difficult to classify instances. That is classifiers are iteratively learned, and misclassified instances are more likely to be chosen in later training sets.

Both of the aforementioned techniques have their advantages and disadvantages. Bagging is trivially parallelizable, and thus more amenable to building a large ensemble, however does not reduce the bias. AdaBoost, on the other hand reduces both variance and bias and has theoretical guarantees, but is sensitive to noise. It is therefore dependent upon the dataset which technique will provide better results. Because of this, Kotsiantis and Pintelas [18] devise a methodology of combining both bagging and AdaBoost in order to create a better ensemble of classifiers.

While the previous techniques modified the training set by altering the distribution of examples, the random subspace method [8] modifies the examples themselves. That is, when attempting to build an ensemble with  $n$  classifiers, each classifier independently chooses a subset of the features to use for training. Thus while most classifiers suffer from the curse of dimensionality, the random subspace method mitigates this by pruning the feature space.

Finally the random forest technique [19] combines the above techniques by using bagging to create the training sets, and then (optionally) applying random subspaces to each training set individually before learning the final classifier. This has the effect of generally producing a highly effective ensemble on most datasets, as combining multiple different sources of randomness leads increased diversity of the training data, and thus increased diversity of the ensemble.

In addition to applying these traditional ensemble methods to the class imbalance problem, modified versions have also been developed. Chawla et al. introduce SMOTE-Boost [5], which combines boosting with SMOTE by, during each iteration of boosting, using SMOTE on the hard to classify instances. Guo and Viktor develop another extension for boosting, DataBoost-IM, which identifies hard instances in order to generate similar synthetic examples, and then reweights the instances to prevent a bias towards the majority class [7]. Liu, Wu, and Zhou propose two methods, EasyEnsemble and BalanceCascade [6], which generate training sets by choosing an equal number of majority and minority class instances from a larger training set. Finally Hido and Kashima introduce a variant of bagging, "Roughly Balanced Bagging" (RB bagging) [20] which alters bagging to emphasize the minority class.

In addition to ensemble methods to deal with the class imbalance problem, sampling techniques can also be employed. The two simplest sampling techniques are oversam-

pling and undersampling. In oversampling, the minority class instances are replicated to create a larger minority class set. Conversely, in undersampling majority class instances are removed in order to level the class distribution. Both of these techniques, among others, have been widely employed [21–23]

A more sophisticated technique which seeks to combat class imbalance is known as SMOTE (Synthetic Minority Over-sampling Technique) [24]. When applying SMOTE, the training set is altered by adding more minority class examples in order to force the class distribution to become more balanced. Instead of simply oversampling the minority class (i.e., duplicating minority examples at random), SMOTE creates new synthetic minority examples by first selecting a minority example and its  $k$  nearest neighbors. The synthetic example is then created by choosing a neighbor and an feature. As the two examples form a line segment in the chosen feature space, the synthetic example's value for the feature is chosen to lie along that line segment.

## 7 Conclusion

We proposed an extension to the RSM to mitigate the effects of class imbalance, called RSM+Sampling, proposing SMOTE be used as the default sampling method. We then compared this method and RSM+usamp against bagging, AdaBoost, random forest, and the random subspace method. We posited that by applying SMOTE to subspaces and then learning classifiers will lead to an improved performance due to more diverse classifiers as well as less noise imputation due to SMOTE. The latter arises as SMOTE is only applied to a much reduced set of features at a time, and is thus a more controlled phenomenon. It is also not affected by the data sparsity and high dimensionality. To test this hypothesis we ran experiments on 21 widely available imbalanced datasets.

The results on the selected datasets showed RSM+SMOTE outperformed all other classifiers tested at the 95% confidence level, and only did not outperform RSM+usamp at the 99% confidence levels. From these results it is apparent that RSM+SMOTE is well suited to overcoming the class imbalance problem. We argue that this is due to the diversity that SMOTE adds to the ensemble. That is SMOTE adds perturbations to the training data based on the features which has a positive effect on the diversity of the ensemble, and therefore increases performance. Based on this and the statistical significance tests, we recommend the use of RSM+SMOTE on imbalanced datasets.

## Acknowledgements

Work was supported in part by the NSF Grant ECCS-0926170 and the Notebaert Premier Fellowship.

## References

1. Chawla, N.V., Cieslak, D.A., Hall, L.O., Joshi, A.: Automatically Countering Imbalance and Its Empirical Relationship to Cost. *Data Mining and Knowledge Discovery* **17**(2) (2008) 225 – 252

2. Dietterich, T.G.: Ensemble methods in machine learning. *Lecture Notes in Computer Science* **1857** (2000) 1–15
3. Breiman, L.: Bagging predictors. *Machine Learning* **24**(2) (1996) 123–140
4. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: *Thirteenth International Conference on Machine Learning*. (1996)
5. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: Smoteboost: improving prediction of the minority class in boosting. In: *In Proceedings of the Principles of Knowledge Discovery in Databases, PKDD-2003*. (2003) 107–119
6. Liu, X.Y., Wu, J., Zhou, Z.H.: Exploratory under-sampling for class-imbalance learning. In: *ICDM '06: Proceedings of the Sixth International Conference on Data Mining, Washington, DC, USA, IEEE Computer Society* (2006) 965–969
7. Guo, H., Viktor, H.L.: Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *SIGKDD Explor. Newsl.* **6**(1) (2004) 30–39
8. Ho, T.: The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence* **20**(8) (1998) 832–844
9. Chawla, N.V., Sylvester, J.: Exploiting diversity in ensembles: Improving the performance on unbalanced datasets. In Haindl, M., Kittler, J., Roli, F., eds.: *MCS. Volume 4472 of Lecture Notes in Computer Science.*, Springer (2007) 397–406
10. Cieslak, D.A., Chawla, N.V.: Learning decision trees for unbalanced data. In: *European Conference on Machine Learning, Springer* (2008)
11. Asuncion, A., Newman, D.: *UCI machine learning repository* (2007)
12. Chang, C.C., Lin, C.J.: *LIBSVM: a library for support vector machines*. (2001) Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
13. DeGroot, M., Schervish, M.: *Probability and Statistics. Third edition edn.* Addison-Wesley (2001)
14. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* **7** (January 2006) 1–30
15. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles. *Machine Learning* **51** (2003) 181–207
16. T.G., D.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning* **40** (August 2000) 139–157(19)
17. Hansen, L.K., Salamon, P.: Neural network ensembles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **12**(10) (1990) 993–1001
18. Kotsiantis, S., Pintelas, P.: Combining bagging and boosting. *International Journal of Computational Intelligence* **1**(4) (2004) 324–333
19. Breiman, L.: Random forests. *Machine Learning* **45**(1) (October 2001) 5–32
20. Hido, S., Kashima, H.: Roughly balanced bagging for imbalanced data. In: *SDM, SIAM* (2008) 143–152
21. Drummond, C., Holte, R.C.: C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. In: *ICML Workshop on Learning from Imbalanced Datasets II*. (2003)
22. Weiss, G.M., Provost, F.: Learning when training data are costly: the effect of class distribution on tree induction. *J. Artif. Int. Res.* **19**(1) (2003) 315–354
23. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.* **6**(1) (2004) 20–29
24. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* **16** (2002) 321–357