Generative Oversampling for Mining Imbalanced Datasets

Alexander Liu, Joydeep Ghosh *Member, IEEE*, and Cheryl Martin *Member, IEEE*

***Abstract*— One way to handle data mining problems where class prior probabilities and/or misclassification costs between classes are highly unequal is to resample the data until a new, desired class distribution in the training data is achieved. Many resampling techniques have been proposed in the past, and the relationship between resampling and cost-sensitive learning has been well studied. Surprisingly, however, few resampling techniques attempt to create new, artificial data points which generalize the known, labeled data. In this paper, we introduce an easily implementable resampling technique (generative oversampling) which creates new data points by learning from available training data. Empirically, we demonstrate that generative oversampling outperforms other well-known resampling methods on several datasets in the example domain of text classification.**

I. Introduction

The imbalanced dataset problem is a special type of classification problem where the class priors are highly unequal and imbalanced. For example, in the simplest two-class case, a balanced problem would have the class priors of both classes approximately equal to each other. In comparison, in an imbalanced problem, one class (the majority class) has a much larger prior probability than the second class (the minority class). Many important real-life two-class problems have minority class priors well under 0.1.

For such problems, it is also often true that it is more important to correctly classify the minority class. In other words, there is a higher cost for misclassifying the minority class than misclassifying the majority class. Example problems that exhibit both class imbalance and a higher misclassification cost for the minority class include detecting cancerous cells, fraud detection [1] [2], keyword extraction [3], oil-spill detection [4], direct marketing [5], and information retrieval [6].

Without accounting for imbalanced priors, a classifier may learn to always predict the majority class. Given that the cost of misclassifying minority class points may be extremely high, a classifier that always predicts the majority class is not acceptable. Many approaches have been proposed and studied in order to handle imbalanced problems (see [7] for a recent survey). One class of solutions is to resample the training set. Resampling changes the priors in the training set by either increasing points from the minority class or decreasing points from the majority class. Other example techniques of dealing with class imbalance include appropriate feature selection [8] [9], one-class learners [10], and cost-sensitive learners which explicitly take misclassification costs into account when learning [11].

This paper focuses on resampling for a variety of reasons. First, empirically, results using resampling have been shown to be competitive [12] or even nearly identical (for certain classifiers) [13] to results using cost-sensitive learning. Second, studies have shown the exact theoretical connection between certain resampling methods and cost-sensitive learning [14] [11]. Moreover, by resampling correctly, any classification algorithm can be made cost-sensitive without having to change the internal workings of the classifier itself [11]. Finally, resampling methods are also extremely simple to implement in practice.

This paper introduces a resampling technique called generative oversampling where artificial data points are generated from an assumed probability distribution whose parameters are learned from the training data. Generative oversampling takes advantage of the fact that in many domains, appropriate families of probability distributions are known that can model data well. Generative oversampling is simple and straightforward to implement. Despite this simplicity, we empirically show that generative oversampling can outperform popular resampling techniques.

II. Related Work

A number of resampling methods have been proposed and studied in the past. Resampling methods can be divided into two categories: oversampling methods and undersampling methods. Oversampling methods balance training class priors by increasing the number of minority class data points, while undersampling methods balance training class priors by decreasing the number of majority class data points. Some widely used approaches are random oversampling, SMOTE [15], random undersampling, and cost-proportionate rejection sampling [11].

Random oversampling increases the number of minority class data points in the training set by randomly replicating existing minority class members. While simplistic, random oversampling has performed well in empirical studies (e.g., [16]) even when compared to other, more complicated oversampling methods. Unfortunately, since random oversampling only replicates existing data points, it has been argued that random oversampling does not add any actual data to the training set.

SMOTE (Synthetic Minority Oversampling TEchnique), an alternative oversampling method, attempts to add information to the training set. Instead of replicating existing data points, "synthetic" minority class members are added to the training set by creating new data points. Empircally, SMOTE has shown to perform well against random oversampling

Alexander Liu and Joydeep Ghosh are with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78713, USA (email: {aliu, ghosh} [at] ece.utexas.edu).

Cheryl Martin is with the Applied Research Laboratories, University of Texas at Austin, Austin, TX 78713, USA (email: cmartin [at] arlut.utexas.edu).

([15], [16]). In SMOTE, a new data point is created from an existing data point as follows: find the n nearest neighbors to the existing data point; randomly select one of the n nearest neighbors; the new, synthetic point is a randomly chosen point on the line segment joining the original data point and its randomly chosen neighbor. In this paper, we introduce an oversampling method which also adds information to the training set by creating synthetic minority class points. Our method requires the choice of a probability distribution to model the minority class.

Random undersampling decreases the number of majority class data points by randomly eliminating majority class data points currently in the training set. Like random oversampling, random undersampling has empirically performed well despite its simplicity [17]. A disadvantage of undersampling is that it removes potentially useful information from the training set. For example, since it indiscriminately removes points, it does not consider the difference between points close to the potential decision boundary and points very far from the decision boundary.

A more sophisticated undersampling method with nice theoretical properties is cost-proportionate rejection sampling [11]. Cost-proportionate rejection sampling is based on a theorem that describes how to turn any classifier which reduces the number of misclassification errors into a cost-sensitive classifier. Given that each data point has a misclassification cost c, each data point in the training set has probability c/z of being included in the resampled training set, where z is a user-defined parameter (e.g., let z equal the largest value of c in the training set in order to maximize the expected size of the resampled dataset).

Interestingly, under certain conditions, random undersampling and cost-proportionate rejection sampling are equivalent methods. Instead of specifying a separate cost c per data point, many cost-sensitive approaches assume a constant cost for misclassifying one class as another. That is, for a k-class problem, a k by k cost-matrix $\mathbf{C}$ can be defined where $\mathbf{C}(i,j)$ is the cost of misclassifying a point that is truly from class i as a point from class j. In the standard framework for studying imbalanced data, k is equal to 2 (the minority class and the majority class), and there is some constant cost for misclassifying the minority class ($\mathbf{C}(1,2)$) and a constant cost for misclassifying the majority class ($\mathbf{C}(2,1)$). While the exact costs may be unknown, it is known that $\mathbf{C}(1,2) > \mathbf{C}(2,1)$. In this case, cost-proportionate rejection sampling boils down to random undersampling with some probabilistically determined rate of resampling. If we seek to maximize the size of the resampled dataset by setting z equal to $\mathbf{C}(1,2)$, then the expected number of majority class points will be $\frac{\mathbf{C}(2,1)}{\mathbf{C}(1,2)}$ times the size of the original majority class. Thus, under common assumptions when empirically classifying imbalanced datasets, cost-proportionate rejection sampling is equivalent to random undersampling with a probabilistically determined number of points to remove from the majority class.

III. Generative Oversampling

In this section, we introduce the generative oversampling algorithm. We first discuss why the use of a probability distribution with resampling is natural and well motivated. We then describe the generative oversampling algorithm which creates completely new, artificial data points via a chosen probability distribution. Finally, we make some comments and observations about generative oversampling.

A. Motivation

For the sake of discussion, consider the following two-class classification problem where the data set X (as well as specific training set X_{train} and test set X_{test}) is the union of points from sets P and Q where:

1) P is a set of points from one class; these points are drawn from some unknown distribution
2) Q is a set of points from the second class; these points are drawn from a second unknown distribution
3) and $\lambda = \frac{|P|}{|X|}$; that is, λ is the prior probability that a point is from set P.

The distributions that generate P and Q can be arbitrarily complicated, and the goal of a two-class classifier is to learn to distinguish between points from P and points from Q. In many domains, the family of probability distributions suitable for modeling P and Q is known. For example, many low-dimensional datasets can be modeled by mixtures of Gaussian distributions ([18]), whereas text datasets can be modelled as mixtures of multinomial distributions ([19]).

A two-class, imbalanced dataset problem can be phrased in terms of P and Q by letting P represent points from the minority class and Q represent points from the majority class. X becomes imbalanced when the minority class prior λ is much less than $1-\lambda$. The goal of resampling is to either increase the number of points drawn from the distribution that produces P or decrease the number of points drawn from the distribution that gives rise to Q. That is, ideally, oversampling techniques [1] would add points to training set X_{train} that have been drawn directly from the distribution that originally produced P.

Unfortunately, we typically cannot draw additional points from this original distribution since it is unknown. Instead, some facsimile for drawing from this unknown distribution is used instead (such as random oversampling or SMOTE). However, while we cannot obtain new points from the original distribution, we can attempt to model the distribution that produced P and create new points from this model. We describe such an algorithm in the next section.

B. Generative oversampling

Generative oversampling can be used in any domain where there exist probability distributions that model the actual data distributions well. Generative oversampling works as follows:

[1] For a discussion of undersampling and its effect on the training set in terms of P and Q, see [11].

1) a probability distribution is chosen to model the minority class
2) based on the training data, parameters for the probability distribution are learned
3) artificial data points are added to the resampled data set by generating points from the learned probability distribution until the desired number of minority class points in the training set has been reached.

The idea behind generative oversampling is simple, straightforward, and is a natural approach in data mining and machine learning. Indeed, the idea of creating artificial data points through a probability distribution with learned parameters has been used for many other applications (e.g., [18] for creating diverse classifier ensembles, [20] for model compression). Surprisingly, however, it has not been used with respect to resampling techniques.

C. Generative oversampling for text datasets

In this paper, we empirically study resampling methods in text classification. The multinomial distribution has been successfully used with the bag of words representation in text mining (e.g., [19]). Thus, we now present a specific example of generative oversampling for text datasets that assumes a multinomial distribution. While other distributions have been applied to model text, we choose the multinomial distribution in our experiments because it has been well studied and because multinomial naive Bayes has been widely applied with success. A multinomial model with Laplace smoothing is learned from the minority class training data such that the probability of a word appearing in a document from the minority class is given by:

$$p(w_i|x_j \in P_{train}) = \frac{l + \sum_{x_j \in P_{train}} x_j(i)}{\sum_{a=1}^{n_v}(l + \sum_{x_b \in P_{train}} x_b(a))} \quad (1)$$

where P_{train} represents the set of minority class points in the training set, w_i represents the ith word in the vocabulary, n_v represents the number of words in the vocabulary, x_j represents the jth document in P_{train}, $x_j(i)$ is the number of times w_i occurs in document x_j, and l controls the amount of smoothing. Note that, if $l > 0$, artificial minority class documents created using generative oversampling have a non-zero probability of containing words not contained in the minority class during training.

An artificial document is created by drawing from the learned multinomial distribution n_{ave} times, where n_{ave} is the average length of the minority class documents in the training set.

D. Discussion

One possible drawback to generative oversampling is that there may not be enough data to properly learn the parameters of the chosen probability distribution. This is related to the difference between relative scarcity and absolute scarcity, two problems that are often conflated when dealing with class imbalance. Absolute scarcity describes a problem where there is simply a small number of data points from a particular class, whereas relative scarcity describes the problem where the prior probability of a class is small relative to the priors of the other classes in the dataset. Generative oversampling can handle classes that are relatively scarce but not absolutely scarce. However, absolute scarcity is a problem for not only generative oversampling, but other techniques as well; that is, handling data that is absolutely scarce is an important and open research question that is not addressed well by current approaches.

Generative oversampling for text using a multinomial distribution also has the following two amenable properties as long as some smoothing is used (i.e., $l > 0$). Firstly, generative oversampling has the capability of adding new minority points outside of the convex hull inscribing the original minority class points in the training set. This is desirable in many cases; for example, a well known problem with SVMs and imbalanced data is that the separating hyperplane is often pushed towards the minority class. By expanding the size of the minority class convex hull, the separating hyperplane is shifted away from the minority class. Secondly, when using generative oversampling, new points can include words not seen in the minority class points in the training set, a feature which discourages overfitting the minority class. The capabilities to expand the minority class convex hull and to allow new, resampled points to take on new words not seen in the training set are due to the Laplace smoothing parameter l. In contrast, neither random oversampling, random undersampling, or SMOTE are able to add points outside of the convex hull or allow new words not seen in the minority class points in the training set.

IV. Experiments

In this section, we empirically test generative oversampling against three other resampling methods: random oversampling, random undersampling, and SMOTE. We use SVMs in the domain of text classification in our empirical tests. We show that both resampling (particularly generative oversampling and random undersampling) and adjusting the tradeoff between the size of the margin and the number of support vectors can improve the performance of SVMs when classifying imbalanced data. Moreover, results obtained by tuning the tradeoff between the margin and support vectors can be improved even further through generative oversampling. We empirically show that generative oversampling works best when used with Laplace smoothing. Finally, we show that generative oversampling is robust to choosing how many resampled data points to add to the training set whereas the performance of other resampling methods (particularly random undersampling) can be highly dependent on the degree of resampling.

Below, we first describe the datasets used in our experiments, then describe our general experimental setup, and finally discuss our results.

A. Datasets

We test our resampling methods using text mining as an example domain. We use a variety of text datasets from various sources. The text datasets come from several past studies

TABLE I
DATASET CHARACTERISTICS

Dataset	Num min class pts	Min class prior(λ)
hitech	116	0.0504
k1b	60	0.0256
la12	521	0.0830
ohscal	709	0.0635
reviews	137	0.0337
sports	122	0.0142

in information retrieval including TREC (http://trec.nist.gov), OHSUMED [21], and WebAce [22]. The datasets from TREC are all newspaper stories from either the LA Times (la12 dataset) or San Jose Mercury (hitech, reviews, sports datasets) classified into different topics. The ohscal dataset contains text related to medicine, while the k1b dataset contains documents from the Yahoo! subject hierarchy. All six of these datasets were included in the CLUTO toolkit [23] [2].

All text datasets were converted into a standard bag of words model. In addition, we used TFIDF weighting and normalized each document vector with the L2 norm after resampling. Finally, since the imbalanced dataset problem is typically posed as a two-class problem, we converted each of our text datasets (which have multiple classes) into two-class problems. For each dataset, we chose the smallest class in each dataset as the minority class, and aggregated all other classes to form the majority class.

Details about these datasets are given in Table I, including the number of minority class points in the data and the value of the minority class prior λ.

B. Experimental setup

For each dataset in our empirical evaluation, we apply the following basic steps:

1) Divide the data into training and test sets
2) Resample training data
3) Apply SVM classifier and evaluate

For each dataset, we create ten different training and test splits by randomly selecting 50% of the data using stratified sampling as the training set and the rest as the test set. We test generative resampling with smoothing parameter $l = 1$ against three other resampling methods: random oversampling, random undersampling, and SMOTE [15]. Note that since SMOTE requires a distance metric, we use 1-cosine similarity as the distance metric since 1-cosine similarity has shown to be a good metric for text data [25]. SMOTE also requires a parameter to control the number of nearest neighbors to consider when resampling; we use 5 as in [15].

In addition, we also perform experiments where we compare generative oversampling with different levels of smoothing. In particular, we try parameter $l = 1$, $l = 0$ (no smoothing), and $l = 5$.

[2]We use the datasets available at http://www.ideal.ece.utexas.edu/data/docdata.tar.gz, which have some additional preprocessing as described in [24].

When resampling, one controls the percentage of the training set comprised of minority class points. We will call this the resampled minority class training prior. A resampled minority class training prior of p% means that after resampling, p% of the training set is comprised of minority class points. In our experiments, we vary the resampled minority class training prior between the natural prior λ and 70%. Note that when the resampled minority class training prior is equal to λ, no actual resampling has been performed.

We use an SVM with a linear kernel, which has been shown to work well in text classification [26]. Specifically, we use the SVM-light implementation by Joachims [27]. Note that, for SVMs, there is an additional parameter used to control the trade-off between the margin and training errors. In SVM-light this is the parameter C [3]. Adjusting C can affect the location of the separating hyperplane. In SVM-light, one can either specify C or use a default value of C estimated from the data. We run experiments using both settings. When specifying C, we perform a search for the best value of C by using a validation set; we further split each training set into a smaller training set (70% of initial training set) and validation set (the remaining 30% of the training set) and search for the best value of C between 2^{-6} and 2^{6}. Note that this tuning is done separately for every experiment (i.e., once for every combination of dataset, training set split, resampled minority class prior, and resampling method).

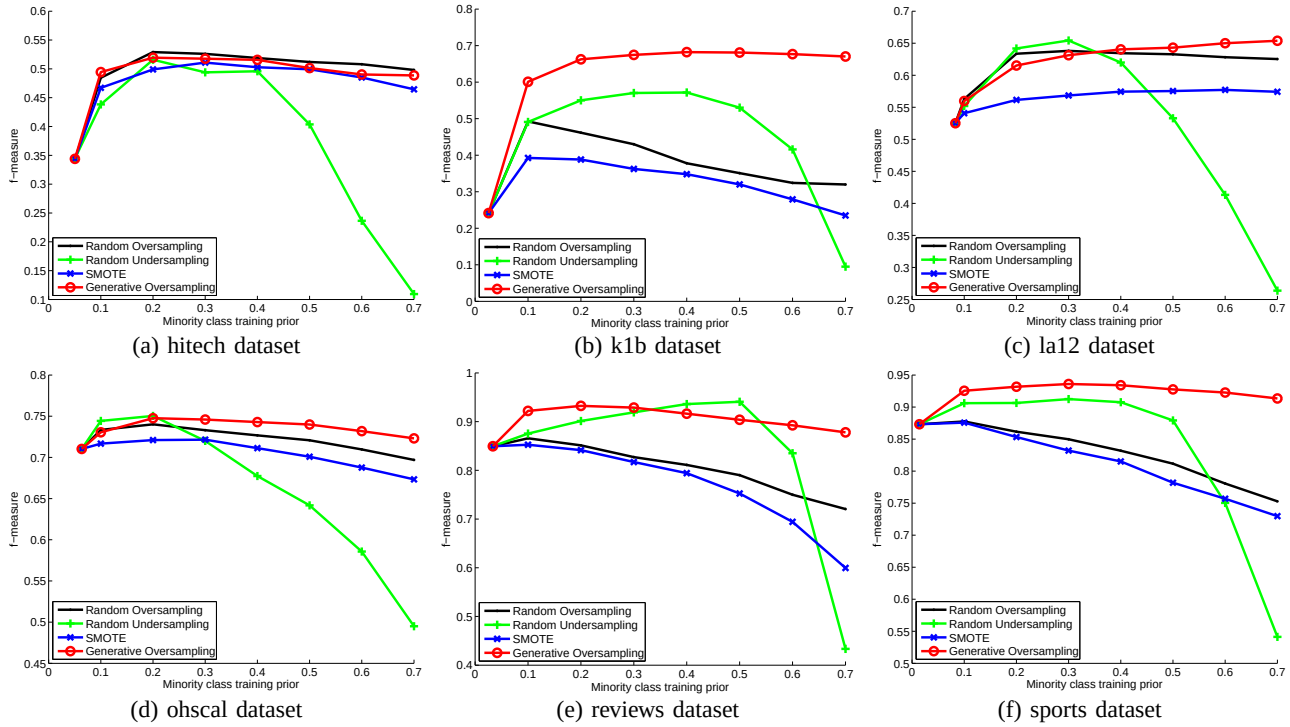
We use f-measure as our evaluation metric. Besides being a popular metric in information retrieval, f-measure has been used in other past studies on imbalanced data (e.g., [9]) and, unlike metrics like accuracy, is considered robust against class imbalance. ROC is another popular metric used to study imbalanced data, but we choose f-measure because of its prevalent and standard use in evaluating text classification.

In summary, we perform three sets of experiments. In the first set, we compare the effect of generative oversampling (with $l = 1$), random oversampling, SMOTE, and random undersampling on SVMs without tuning the trade-off between the size of the margin and the number of training errors. In the second set of experiments, we perform the same experiments while tuning the trade-off between the size of the margin and number of training errors. Finally, in the third set of experiments, we show that generative oversampling works well when used with Laplace smoothing by comparing generative oversampling with $l = 0$, $l = 1$, and $l = 5$.

C. Results

Figure 1 plots our results comparing generative oversampling, random oversampling, SMOTE, and random undersampling on each of the six datasets when all default parameters for linear SVMs are used. Figure 2 shows the results when the tradeoff between margin size and number of training errors is tuned via a validation set. The plots show f-measure as a function of resampled minority class training prior.

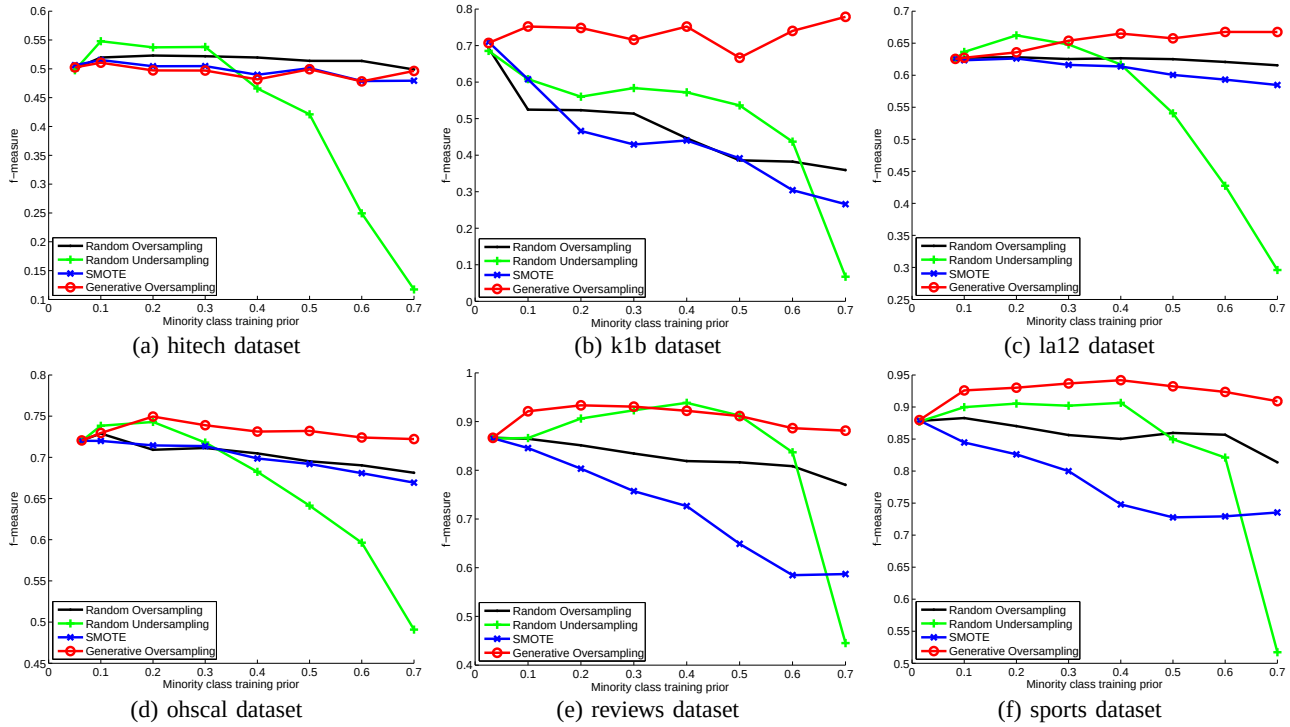
[3]Not to be confused with the cost matrix $\mathbf{C}$ used in section II



(a) hitech dataset (b) k1b dataset (c) la12 dataset

(d) ohscal dataset (e) reviews dataset (f) sports dataset

Fig. 1. f-measure vs minority class training prior



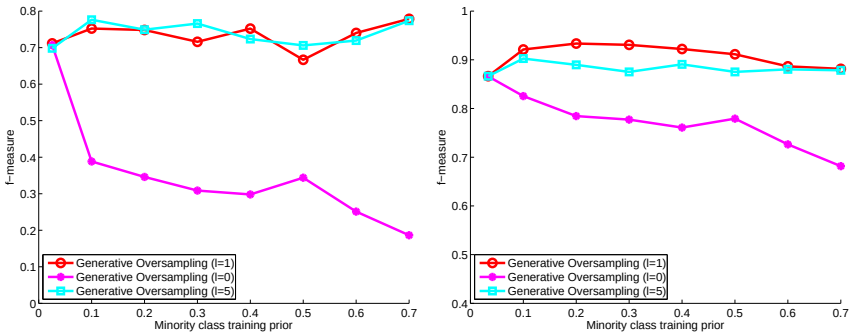
(a) hitech dataset (b) k1b dataset (c) la12 dataset

(d) ohscal dataset (e) reviews dataset (f) sports dataset

Fig. 2. f-measure vs minority class training prior while tuning C

D. Effects on location of the separating hyperplane

Generative oversampling potentially increases the size of the convex hull surrounding the minority class by producing artificial data points that occur both inside and outside of the original convex hull inscribing the minority class points in the training set. Random undersampling potentially reduces the size of the convex hull surrounding majority class training points by removing points from the majority class. Thus, both generative oversampling and random undersampling can be effective at moving the decision boundary found by a linear



(a) k1b dataset (b) reviews dataset

Fig. 3. Example results with different levels of smoothing

SVM, whereas random oversampling and SMOTE merely add points in the existing minority class convex hull.

We see the empirical effects of this in both figures 1 and 2 [4]. In figure 1, where the parameter C has not been tuned, we see that in all but the hitech datasets, either generative oversampling or random undersampling produce the highest f-measure. In particular, generative oversampling is either the clear winner (k1b, sports) or is extremely competitive with random undersampling.

Tuning the parameter C can also affect the location of the separating hyperplane. Thus, we see in figure 2 that the results with the natural prior are much higher than the results with the natural prior without tuning in figure 1. In particular, note that the results using the natural prior (even after tuning C) can be improved upon by using generative resampling or random undersampling. In comparison, there is minimal or no improvement using random oversampling or SMOTE as compared to using the natural prior if C is tuned. In fact, using random oversampling or SMOTE often does worse than simply using the natural prior. Thus, regardless of whether C is tuned, generative oversampling and random undersampling can both be used to improve upon results using the natural prior.

Finally, Figure 3 contains sample results where we compare generative oversampling with different levels of smoothing (note that all of these experiments were run while tuning for C). The results in figure 3 show two general cases which are representative of performance on all the datasets: either results with $l = 5$ and $l = 1$ are comparable, or results with $l = 1$ are better than results with $l = 5$. With $l = 0$, generative oversampling does very poorly (often worse than random oversampling or SMOTE). When $l = 0$, generative oversampling no longer creates points outside of the original convex hull, supporting our argument that the reason that generative oversampling (with $l = 1$) performs well is due to its ability to expand the original convex hull of the minority class. For $l = 5$, the results obtained with generative oversampling are clearly worse than generative oversampling with $l = 1$ on two of the datasets. On the other datasets, generative oversampling with $l = 1$ and $l = 5$ are comparable, although $l = 5$ is slightly better at low minority class training priors. The exact reasons for the difference in performance between $l = 1$ and $l = 5$ is an open question.

[4]Note that the vertical axes of all graphs are scaled differently in order to maximize visualization within each graph; however, to facilitate comparisons, the vertical axis of graphs corresponding to the same dataset are scaled identically.

E. Robustness to choice of minority class training prior after resampling

Unfortunately, the exact minority class training prior to use may be unknown. As many authors have noted in the past, using a completely balanced training set (i.e., minority class training prior of 0.5) rarely (never in our experiments) yields the best results. If there is an exact, known misclassification cost, then previous studies ([14], [11]) offer a guidance on the minority class training prior to use for the best results. However, in many cases, there is no exact notion of misclassification costs, meaning that there is no corresponding guidance as to which minority class training prior to use in practice [13]. Moreover, in many domains, the exact misclassification costs may change over time [28], [29]. While [30] shows there may be guidelines (depending on the evaluation metric being used) for choosing a good minority class training prior, the best minority class training prior varies from dataset to dataset and must be determined experimentally ([30], [31]). Thus, it is desirable for a resampling method to be robust with respect to the resampled minority class training prior.

In our experiments, random undersampling is particularly sensitive to choosing the correct resampled minority class training prior. In all six of our datasets, there is a very clear degradation in f-measure when the minority class training prior is either too low or too high regardless of whether C is tuned. Thus, while the best f-measure obtained using random undersampling is often competitive with results obtained from generative oversampling, the f-measure produced using random undersampling is extremely dependent on choosing the resampled minority class training prior.

Thus, generative resampling performs well compared to random oversampling, SMOTE, and random undersampling with respect to f-measure as well as with respect to robustness to minority class training prior. In contrast, SMOTE and random oversampling often cannot improve over using the

natural prior (i.e., no resampling), particularly when C is tuned. Results with random undersampling depend heavily on choosing the appropriate minority class training prior, which may be difficult in practice.

F. Beyond text classification and SVMs

Finally, we have performed additional, preliminary experiments using different classifiers and using different types of datasets (particularly low-dimensional datasets). In particular, we should note that when using a k-nearest neighbor classifier on the six datasets tested in this paper, SMOTE performs quite well while generative oversampling does not. This seems to be because the bias of SMOTE matches the bias of k-nearest neighbors, whereas the bias of generative oversampling matches the bias of SVMs. That is, SMOTE performs local smoothing on the training set, which matches the k-nearest neighbor classifier which classifies points based on local neighborhoods. In contrast, generative oversampling performs a global smoothing on the training set that increases the size of the minority class convex hull, which, as noted above, is important for SVMs but is much less impactful on k-nearest neighbor classifiers. We plan to publish these and other empirical tests in future work.

V. Conclusion

In this paper, we have introduced the generative oversampling algorithm, a resampling algorithm that creates artificial data points from a probability distribution learned from the minority class. Empirically, we have shown that generative oversampling works well for a range of text classification datasets using linear SVMs. Moreover, if the best minority class training prior is unknown, generative oversampling has the added benefit of producing results which are robust to changes in minority class training prior. Generative oversampling is also simple to implement and can be used on a variety of different data types by selecting appropriate generative models. Therefore, it is a viable and flexible alternative whenever resampling methods are used.

Acknowledgment

The authors would like to thank colleagues at the Applied Research Laboratories and IDEAL for helpful comments and suggestions, including Karl Fisher, Tom Hetheringtom, Dung Lam, and Sara Matzner.

References

[1] P. K. Chan and S. J. Stolfo, "Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection," in *Knowledge Discovery and Data Mining*, 1998, pp. 164–168. [Online]. Available: citeseer.ist.psu.edu/article/chan98toward.html

[2] C. Phua, D. Alahakoon, and V. Lee, "Minority report in fraud detection: Classification of skewed data," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 50–59, 2004.

[3] P. D. Turney, "Learning algorithms for keyphrase extraction," *Information Retrieval*, vol. 2, no. 4, pp. 303–336, 2000.

[4] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine Learning*, vol. 30, no. 2-3, pp. 195–215, 1998.

[5] C. X. Ling and C. Li, "Data mining for direct marketing: Problems and solutions," *Knowledge Discovery and Data Mining*, pp. 73–79, 1998.

[6] D. Lewis and W. Gale, "Training text classifiers by uncertainty sampling," in *Proceedings of the Seventh Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.

[7] S. Visa and A. Ralescu, "Issues in mining imbalanced data sets - a review paper," in *Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference*, 2005, pp. 67–73.

[8] M. D. del Castillo and J. I. Serrano, "A multistrategy approach for digital text categorization from imbalanced documents," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 70–79, 2004.

[9] Z. Zheng, X. Wu, and R. K. Srihari, "Feature selection for text categorization on imbalanced data." *SIGKDD Explorations*, vol. 6, no. 1, pp. 80–89, 2004.

[10] B. Raskutti and A. Kowalczyk, "Extreme re-balancing for svms: a case study." *SIGKDD Explorations*, vol. 6, no. 1, pp. 60–69, 2004.

[11] B. Zadrozny, J. Langford, and N. Abe, "Cost-sensitive learning by cost-proportionate example weighting," in *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, 2003.

[12] K. McCarthy, B. Zabar, and G. Weiss, "Does cost-sensitive learning beat sampling for classifying rare classes?" in *UBDM '05*. New York, NY, USA: ACM Press, 2005, pp. 69–77.

[13] M. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," *ICML*, 2003.

[14] C. Elkan, "The foundations of cost-sensitive learning," *Proc. IJCAI*, 2001.

[15] N. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[16] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explorations*, vol. 6, no. 1, pp. 20–29, 2004.

[17] Zhang and Mani, "knn approach to unbalanced data distributions: A case study involving information extraction," *ICML*, 2003.

[18] P. Melville and R. J. Mooney, "Diverse ensembles for active learning," *Proc. ICML*, pp. 584–591, 2004.

[19] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[20] C. Bucila, R. Caruana, and A. Niculescu-Mizil, "Model compression." in *KDD*, 2006, pp. 535–541.

[21] W. Hersh, C. Buckley, T. J. Leone, and D. Hickam, "Ohsumed: An interactive retrieval evaluation and new large test collection for research," *Proceedings of ACM SIGIR*, pp. pp 192–201, 1994.

[22] E. H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore, "Webace: A web agent for document categorization and exploration," *Proceedings of the Second International Conference on Autonomous Agents*, pp. pp 408–415, 1998.

[23] G. Karypis, "Cluto - a clustering toolkit," *Technical report*, 2002.

[24] S. Zhong and J. Ghosh, "A comparative study of generative models for document clustering," *SDM Workshop on Clustering High Dimensional Data and Its Applications*, 2003.

[25] I. S. Dhillon, J. Fan, and Y. Guan, "Efficient clustering of very large document collections," in *Data Mining for Scientific and Engineering Applications*, V. K. R. Grossman, C. Kamath and R. Namburu, Eds. Kluwer Academic Publishers, 2001, invited book chapter.

[26] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information Retrieval*, vol. 1, no. 1/2, pp. 69–90, 1999.

[27] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proceedings of ECML-98, 10th European Conference on Machine Learning*, C. Nédellec and C. Rouveirol, Eds., no. 1398. Chemnitz, DE: Springer Verlag, Heidelberg, DE, 1998, pp. 137–142.

[28] C. Drummond and R. C. Holte, "Cost curves: An improved method for visualizing classifier performance," *Machine Learning*, vol. 65(1), pp. pp 95–130, 2006.

[29] F. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. pp 445–453, 1998.

[30] G. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *JAIR pp 315-324*, 2003.

[31] A. Estabrooks, T. Jo, and N. Japkowicz, "A multiple resampling method for learning from imbalanced data sets," in *Computational Intelligence*, vol. Volume 20, Number 1, 2004.