Chapter 8
Effects of Oversampling Versus Cost-Sensitive Learning for Bayesian and SVM Classifiers

Alexander Liu, Cheryl Martin, Brian La Cour, and Joydeep Ghosh

Abstract In this chapter, we examine the relationship between cost-sensitive learning and resampling. We first introduce these concepts, including a new resampling method called "generative oversampling," which creates new data points by learning parameters for an assumed probability distribution. We then examine theoretically and empirically the effects of different forms of resampling and their relationship to cost-sensitive learning on different classifiers and different data characteristics. For example, we show that generative oversampling used with linear SVMs provides the best results for a variety of text data sets. In contrast, no significant performance difference is observed for low-dimensional data sets when using Gaussians to model distributions in a naive Bayes classifier. Our theoretical and empirical results in these and other cases support the conclusion that the relative performance of cost-sensitive learning and resampling is dependent on both the classifier and the data characteristics.

8.1 Introduction

Two assumptions of many machine learning algorithms used for classification are that (1) the prior probabilities of all classes in the training and test sets are approximately equal and (2) mistakes on misclassifying points from any class should be penalized equally. In many domains, one or both of these assumptions are violated. Example problems that exhibit both imbalanced class priors and a higher

Alexander Liu · Cheryl Martin · Brian La Cour
Applied Research Labs, University of Texas at Austin, Austin, TX, USA,
e-mail: ayliu@mail.utexas.edu, cmartin@arlut.utexas.edu, blacour@arlut.utexas.edu

Alexander Liu · Joydeep Ghosh
Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA,
e-mail: ayliu@mail.utexas.edu, ghosh@ece.utexas.edu

R. Stahlbock et al. (eds.), *Data Mining,* Annals of Information Systems 8,
DOI 10.1007/978-1-4419-1280-0_8,

misclassification cost for the class with fewer members include detecting cancerous cells, fraud detection [3, 21], keyword extraction [23], oil-spill detection [12], direct marketing [14], information retrieval [13], and many others.

Two different approaches to address these problems are resampling and cost-sensitive learning. Resampling works by either adding members to a class (oversampling) or removing members from a class (undersampling). Resampling is a classifier-agnostic approach and can therefore be used as a preprocessing step requiring no changes to the classification algorithms. In contrast, cost-sensitive learning approaches may modify classifier algorithms to minimize the total or expected cost of misclassification incurred on some test set. Some studies have shown that resampling can be used to perform cost-sensitive learning. However, in this chapter, we contrast resampling methods with cost-sensitive learning approaches that do not use resampling.

We examine the relationship between cost-sensitive learning and two oversampling methods: random oversampling and generative oversampling. We first introduce these concepts, including a new resampling method called "generative oversampling." We compare the performance both theoretically and empirically using a variety of classifiers and data with different characteristics. In particular, we compare low versus high-dimensional data, and we compare Bayesian classifiers and support vector machines, both of which are very popular and widely used machine learning algorithms. Since there is already an abundance of empirical studies comparing resampling techniques and cost-sensitive learning, the emphasis of this chapter is to examine oversampling and its relationship with cost-sensitive learning from a theoretical perspective and to analyze the reasons for differences in empirical performance.

For low-dimensional data, assuming a Gaussian event model for the naive Bayes classifier, we show that random oversampling and generative oversampling theoretically increase the variance of the estimated sample mean compared to learning from the original sample (as done in cost-sensitive naive Bayes). Empirically, using generative oversampling and random oversampling seems to have minimal effect on Gaussian naive Bayes beyond adjusting the learned priors. This result implies that there is no significant advantage for resampling in this context. In contrast, for high-dimensional data, assuming a multinomial event model for the naive Bayes classifier, random oversampling and generative oversampling change not only the estimated priors but also the parameter estimates of the multinomial distribution modeling the resampled class. This conclusion is supported both theoretically and empirically. The theoretical analysis shows that oversampling and cost-sensitive learning are expected to perform differently in this context. Empirically, we demonstrate that oversampling outperforms cost-sensitive learning in terms of producing a better classifier.

Finally, we present parallel empirical results for text classification with linear SVMs. We show empirically that generative oversampling used with linear SVMs provide the best results, beating any other combination of classifier and resampling/cost-sensitive method that we tested on our benchmark text data sets. We then discuss our hypothesis for why generative oversampling in particular works well with SVMs and present experiments to support this hypothesis.

8.2 Resampling

Resampling is a simple, classifier-agnostic method of rebalancing prior probabilities. Resampling has been widely studied, particularly with respect to two-class problems, where the class with the smaller class prior is called the minority class and the class with the larger prior is called the majority class. By convention, the positive class is set as the minority class and the negative class is set as the majority class.

Resampling creates a new training set from the original training set. Many resampling methods have been proposed and studied in the past. Resampling methods can be divided into two categories: oversampling and undersampling. Oversampling methods increase the number of minority class data points, and undersampling methods decrease the number of majority class data points. Some widely used approaches are random oversampling, SMOTE [4], random undersampling, and cost-proportionate rejection sampling [26].

This chapter focuses on two oversampling methods: random oversampling and generative oversampling, introduced below. Some empirical comparisons against SMOTE and random undersampling are also provided for context, but the empirical results in this chapter are primarily used to illustrate analytical results (see [18, 24, 9] for some empirical benchmarks of resampling techniques).

8.2.1 Random Oversampling

Random oversampling increases the number of minority class data points in the training set by randomly replicating existing minority class members. Random oversampling has performed well in empirical studies (e.g., [1]) even when compared to other, more complicated oversampling methods. However, random oversampling has been criticized since it only replicates existing training data and may lead to overfitting [18]. Both SMOTE [4] and generative oversampling address this criticism by creating artificial points instead of replicating existing points.

8.2.2 Generative Oversampling

The set of points in the minority class is characterized by some unknown, true distribution. Ideally, to resample a data set, one could augment the existing data with points drawn from this true distribution until the training set has the desired ratio of positive and negative class points.

Unfortunately, since the true distribution is typically unknown, one cannot usually draw additional points from this original distribution. However, one can attempt to model the distribution that produced the minority class and create new points based on this model. This is the motivation for generative oversampling [15], an

oversampling technique that draws additional data points from an assumed distribution. Parameters for this distribution are learned from existing minority class data points. Generative oversampling could be effective in problem domains where there are probability distributions that model the actual data distributions well. Generative oversampling works as follows:

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.

Generative oversampling is simple and straightforward. The idea of creating artificial data points through a probability distribution with learned parameters has been used in other applications (e.g., [19] for creating diverse classifier ensembles, [2] for model compression). Surprisingly, however, it has not previously been used to address imbalanced class priors.

8.3 Cost-Sensitive Learning

In a "typical" classification problem, there is an equal misclassification cost for each class. Cost-sensitive learning approaches, however, account for conditions where there are unequal misclassification costs. The goal of cost-sensitive learning is to minimize the total or expected cost of misclassification incurred on some test set.

Researchers have looked at a number of ways of modeling costs in cost-sensitive learning. Perhaps the most common approach is to define a cost for classifying a point from class $Y = y_j$ as a point from class $Y = y_i$ [6]. In this formulation, a cost matrix $\mathbf{C}$ can be defined where $\mathbf{C}(i, j)$ is the misclassification cost for classifying a point with true class $Y = y_j$ as class $Y = y_i$. Typically, $\mathbf{C}(i, i)$ is set to zero for all i such that correct classifications are not penalized. In this case, the decision rule is modified (as discussed in [6]) to predict the class that minimizes $\sum_j P(Y = y_j|\mathbf{X} = \mathbf{x})C(i, j)$, where $\mathbf{x}$ is the data point currently being classified. When costs are considered in the two-class imbalanced data set problem, a two-by-two cost matrix can be defined, meaning that all points in the positive class share some misclassification cost and all points in the negative class share some misclassification cost.

Different classifiers can be modified in different ways in order to take costs into account. For example, a Bayesian classifier can be easily modified to predict the class that minimizes $\sum_j P(Y = y_j|\mathbf{X} = \mathbf{x})C(i, j)$ as in Section 8.5.2. This modification involves shifting the decision boundary by some threshold. In comparison, SVMs can be modified as described in [20], where instead of a single parameter controlling the number of empirical errors versus the size of the margin, a separate parameter for false positives and a second parameter for false negatives are used.

8.4 Related Work

In [6], a direct connection between cost-sensitive learning and resampling is made. The author shows that, theoretically, one can resample points at a specific rate in order to accomplish cost-sensitive learning. In Section 4.1 of [6], the author describes the effect of resampling on Bayesian classifiers. In particular, the author claims that resampling only changes the estimates of the prior probabilities. Thus, an equivalent model can be trained either through resampling or a cost-sensitive Bayesian model. In this chapter, we further examine the assumptions required for this equivalence to hold. The remainder of this section describes additional related work.

Widely used resampling approaches include SMOTE [4], random oversampling, random undersampling, and cost-proportionate rejection sampling [26]. 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 [27]. 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 [26]. Cost-proportionate rejection sampling is based on a theorem that describes how to turn any classifier that reduces the number of misclassification errors into a cost-sensitive classifier. Given that each data point has a misclassification cost, each data point in the training set has a probability of being included in the resampled training set proportional to that point's misclassification cost. We limit the scope of analysis in this chapter to oversampling methods, but a discussion of the relationship between cost-proportionate rejection sampling and cost-sensitive learning is provided in [26].

SMOTE (Synthetic Minority Oversampling TEchnique), an 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. A new data point is created from an existing data point as follows: find the k nearest neighbors to the existing data point ($k = 5$ in [4] and in this chapter); randomly select one of the k 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. Empirically, SMOTE has been shown to perform well against random oversampling [4, 1]. Compared to generative oversampling (a parametric oversampling method that adds synthetic points), SMOTE can be considered a non-parametric method.

Empirically, there seems to be no clear winner as to which resampling technique to use or whether cost-sensitive learning outperforms resampling [16, 18, 24, 9]. Many studies have been published, but there is no consensus on which approach is generally superior. Instead, there is ample empirical evidence that the best resampling method to use is dependent on the classifier [9]. Since there is already an abundance of empirical studies comparing resampling techniques and cost-sensitive

learning, the emphasis of this paper is to examine oversampling and its relationship with cost-sensitive learning from a theoretical perspective and to analyze the reasons for differences in empirical performance.

8.5 A Theoretical Analysis of Oversampling Versus Cost-Sensitive Learning

In this section, we study the effects of random and generative oversampling on Bayesian classification and the relationship to a cost-sensitive learning approach. We begin with a brief review of Bayesian classification and discuss necessary background. We then examine two cases and analyze differences in the estimates of the parameters that must be calculated in each case to estimate the probability distributions being used to model the naive Bayes likelihood. For the first case, with a Gaussian data model, we show that there is little difference between random oversampling, generative oversampling, and cost-sensitive learning. When multinomial naive Bayes is used, however, there is a significant difference. In this case, we show that the parameter estimates one obtains after either random or generative oversampling differ significantly from the parameter estimates used for cost-sensitive learning.

8.5.1 Bayesian Classification

Suppose one is solving a two-class problem using a Bayesian classifier. Let us denote the estimated conditional probability that some (possibly multi-dimensional) data point $\mathbf{x}$ is from the positive class y_+ given $\mathbf{x}$ as $\hat{P}(Y = y_+|\mathbf{X} = \mathbf{x})$ and the estimated conditional probability that $\mathbf{x}$ is from the negative class y_- given $\mathbf{x}$ as $\hat{P}(Y = y_-|\mathbf{X} = \mathbf{x})$. According to Bayes rule:

$$\hat{P}(Y = y_+|\mathbf{X} = \mathbf{x}) = \frac{\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+)\hat{P}(Y = y_+)}{\hat{P}(\mathbf{X} = \mathbf{x})} \tag{8.1}$$

and

$$\hat{P}(Y = y_-|\mathbf{X} = \mathbf{x}) = \frac{\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-)\hat{P}(Y = y_-)}{\hat{P}(\mathbf{X} = \mathbf{x})} \tag{8.2}$$

where $\hat{P}(Y = y_+)$ and $\hat{P}(Y = y_-)$ are the class priors and $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+)$, $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-)$, $\hat{P}(Y = y_+)$, and $\hat{P}(Y = y_-)$ are estimated from the training set. $\hat{\theta}_+$ and $\hat{\theta}_-$ are the estimates of the parameters of the probability distributions being used to model the likelihoods $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+)$ and $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-)$. The decision rule is to assign $\mathbf{x}$ to y_+ if the posterior probability $\hat{P}(Y = y_+|\mathbf{X} = \mathbf{x})$ is greater than or equal to $\hat{P}(Y = y_-|\mathbf{X} = \mathbf{x})$. This is equivalent to classifying $\mathbf{x}$ as y_+ if

$$\frac{\hat{P}(Y = y_+|\mathbf{X} = \mathbf{x})}{\hat{P}(Y = y_-|\mathbf{X} = \mathbf{x})} \geq 1. \tag{8.3}$$

8.5.2 Resampling Versus Cost-Sensitive Learning in Bayesian Classifiers

More generally, one can adjust the decision boundary by comparing the ratio of the two posterior probabilities to some constant. That is, one can adjust the decision boundary by assigning $\mathbf{x}$ to y_+ if

$$\frac{\hat{P}(Y = y_+|\mathbf{X} = \mathbf{x})}{\hat{P}(Y = y_-|\mathbf{X} = \mathbf{x})} \geq \alpha, \tag{8.4}$$

where α is used to denote some constant. For example, one may use a particular value of α if one has known misclassification costs [6]. If one were to use a cost-sensitive version of Bayesian classification, α is based on the cost c_+ of misclassifying a positive class point and the cost c_- of misclassifying a negative class point. In this case, $\alpha = c_-/c_+$ based on the cost-sensitive decision rule given in Section 8.3.

One can also adjust the learned decision boundary by resampling (i.e., adding or removing points from the training set) which changes the estimated priors of the classes. Let $\hat{P}^{(rs)}(Y = y_+)$ and $\hat{P}^{(rs)}(Y = y_-)$ denote the estimated class priors after resampling (regardless of what resampling method has been used). Let $\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$ and $\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})$ be estimated from the resampled training set. If $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$ and $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})$, then the effect of using $\alpha \neq 1$ and the effect of adjusting the priors by resampling can be made exactly equivalent. That is, if resampling only changes the learned priors, then resampling at a specific rate corresponding to $\alpha = c_-/c_+$ is equivalent to cost-sensitive learning. In particular, one can show that if $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$ and $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})$, then resampling to adjust the priors to correspond to $\alpha = c_-/c_+$ can be accomplished if

$$\alpha = \frac{c_-}{c_+} = \frac{\hat{P}^{(rs)}(Y = y_-)\hat{P}(Y = y_+)}{\hat{P}^{(rs)}(Y = y_+)\hat{P}(Y = y_-)}. \tag{8.5}$$

However, in practice, this equivalency may not be exact since resampling may do more than simply adjust the class priors. That is, the assumption that $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$ and $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-) = \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})$ may be invalid because the estimated parameters with and without resampling may change.

In the remainder of this section, we will theoretically examine the effect of two resampling techniques (random oversampling and generative oversampling) on probability estimation and Bayesian learning. In particular, we will examine the difference between learning from the resampled set and learning from the original training set when Gaussian and multinomial distributions are chosen to model the resampled class.

We will assume without loss of generality that the positive class is being oversampled. In this case, since points are neither added nor removed from the negative class, $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_-,\hat{\theta}_-)=\hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_-,\hat{\theta}_-^{(rs)})$. Thus, we will examine how $\hat{\theta}_+$ may differ from $\hat{\theta}_+^{(rs)}$ due to oversampling. Since we will only be discussing the positive class, we will omit the $+$ subscripts when it is obvious that we are referring to parameters estimated on the positive class.

In addition, we will also use the notation $\hat{\theta}^{(r)}$ and $\hat{\theta}^{(g)}$ to refer to the parameters estimated after random oversampling and generative oversampling, respectively, when such a distinction needs to be made, while $\hat{\theta}^{(rs)}$ will continue to refer to parameter estimates after either resampling technique has been used, and $\hat{\theta}$ will continue to refer to parameters estimated from the original training set when no resampling has occurred.

8.5.3 Effect of Oversampling on Gaussian Naive Bayes

In this section, we examine the effect of oversampling when $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+)$ and $\hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+^{(rs)})$ are modeled by Gaussian distributions. In Gaussian naive Bayes, each feature is modeled by an independent Gaussian distribution. Thus, $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+)=\prod_{i=1}^{d}N(x_i|\mu_{+,i},\sigma_{+,i})$ where d is the number of dimensions, $\mu_{+,i}$ and $\sigma_{+,i}$ are the mean and standard deviation estimated for the ith dimension of the positive class, and $N(x_i|\mu_{+,i},\sigma_{+,i})$ is the probability that a normal distribution with parameters $\mu_{+,i}$ and $\sigma_{+,i}$ generated x_i. Since we are only discussing oversampling the positive class, we will drop the $+$ subscripts and simply refer to the parameters as $\hat{\theta}$, μ_i, and σ_i.

For the sake of simplicity, we will limit our discussion to one-dimensional Gaussian distributions. However, since the parameters of each dimension are estimated independently in a naive Bayes classifier, our analysis can be extended to multiple dimensions if the features are indeed independent. Analysis when features are correlated will be left to future work. In the one-dimensional case, $\hat{\theta}$ corresponds to a single sample mean and sample standard deviation estimated from the positive class points in the original training set, while $\hat{\theta}^{(rs)}$ corresponds to the sample mean and sample standard deviation estimated after resampling.

We will first examine the theoretical effect of oversampling on estimating $\hat{\theta}$, $\hat{\theta}^{(r)}$, and $\hat{\theta}^{(g)}$. For the sake of brevity, we limit our discussion of these parameter estimates to the expected value and variance of the sample mean. In particular, we show that the expected value of the sample mean is always the same regardless of whether no resampling, random oversampling, or generative oversampling is applied.

Let the set of n points in the positive class be denoted as $X_1,\ldots,X_n$. These points are an i.i.d. set of random variables from a normal distribution with true mean μ and true variance σ^2. Both μ and σ^2 are unknown and must be estimated as parameters $\hat{\theta}$.

The sample mean estimated from the original training set will be denoted as $\bar{X}$ while the sample mean for the training set created after resampling will be de-

noted as either $\bar{X}_*^{(r)}$ for random oversampling or $\bar{X}_*^{(g)}$ for generative oversampling. Note that, in the Appendix, we make a differentiation between the sample mean estimated only on the newly resampled points (denoted as $\bar{X}^{(r)}$ for random oversampling) versus the sample mean estimated for a training set comprised of both the resampled points and the original points (denoted as $\bar{X}_*^{(r)}$). Here, however, we will simply present results for the "pooled" training set consisting of both the resampled points and the original points.

Consider the sample mean of the original sample, $\bar{X}$. The expected value and variance of the sample mean are as follows:

$$E[\bar{X}] = \mu \tag{8.6}$$

$$\text{Var}[\bar{X}] = \frac{\sigma^2}{n}. \tag{8.7}$$

In the next sections, we will find the expected value and variance of the sample mean of the points generated by random oversampling and generative oversampling. Derivations of these equations can be found in the Appendix.

8.5.3.1 Random Oversampling

Consider a random sample $X_1^{(r)}, \ldots, X_m^{(r)}$ produced through random oversampling. Thus, each $X_j^{(r)} = X_{K_j}$, where $K_1, \ldots, K_m$ are i.i.d. uniformly distributed random variables over the discrete set $\{1, \ldots, n\}$.

We now seek the mean and variance of $\bar{X}_*^{(r)}$.

For the mean, we have

$$E[\bar{X}_*^{(r)}] = \mu \tag{8.8}$$

and

$$\text{Var}[\bar{X}_*^{(r)}] = \left[1 + \frac{m(n-1)}{(n+m)^2}\right] \frac{\sigma^2}{n}. \tag{8.9}$$

Thus, the expected value of the sample mean of the pooled training set is equal to the expected value of the sample mean without resampling. However, the variance of the estimated sample mean of the training set after resampling is greater.

8.5.3.2 Generative Oversampling

Now consider points $X_1^{(g)}, \ldots, X_m^{(g)}$ created via generative oversampling. These points are of the form

$$\mathbf{X}_j^{(g)} = \bar{\mathbf{X}} + s\mathbf{Z}_j, \tag{8.10}$$

where $X_j^{(g)}$ is the j^{th} point created by generative oversampling, s is the original estimated sample standard deviation, and $Z_1, \ldots, Z_m$ are i.i.d. $N(0,1)$ and independent of $X_1, \ldots, X_n$ as well.

The expected value of the mean $\bar{X}_*^{(g)}$ is

$$E[\bar{X}_*^{(g)}] = \mu, \tag{8.11}$$

while the variance of the sample mean is

$$\mathrm{Var}[\bar{X}_*^{(g)}] = \left[1 + \frac{mn}{(n+m)^2}\right] \frac{\sigma^2}{n}. \tag{8.12}$$

Thus, like random oversampling, the sample mean estimated from the resampled points created via generative oversampling has, on average, the same value as the sample mean estimated from the original points, but with greater variance.

8.5.3.3 Comparison to Cost-Sensitive Learning

A cost-sensitive naive Bayes classifier uses the parameter estimates $\hat{\theta}$ from the original set of points. Thus, in expectation, the estimated mean for a Gaussian naive Bayes classifier will be the same, regardless of whether random oversampling, generative oversampling, or cost-sensitive learning is used. When one resamples, one incurs additional overhead in terms of time required to create additional samples, memory needed to store the additional samples, and additional time required to train on the resampled points. Cost-sensitive naive Bayes is therefore preferable over resampling when a Gaussian distribution is assumed. We will support this claim empirically in Section 8.6.2.

8.5.4 Effects of Oversampling for Multinomial Naive Bayes

In multinomial naive Bayes (see [17] for an introduction to multinomial naive Bayes), there is a set of d possible features, and the probability that each feature will occur needs to be estimated. For example, in the case of text classification, each feature is a word in the vocabulary, and one needs to estimate the probability that a particular word will occur. Thus, the parameter vector θ for a multinomial distribution is a d-dimensional vector, where θ_k is the probability that the kth word in the vocabulary will occur.

Let F_i denote the number of times the ith word occurs in the positive class in the training set, and let $F_i^{(r)}$ represent the number of times that a word occurs in only the randomly oversampled points (we will use $F_i^{(g)}$ when discussing generative oversampling). In addition, let n represent the number of words that occur in the positive class in the training set, and let m represent the number of words that occur in the resampled points.

For the case where there are no resampled points in the training set, the maximum likelihood estimator for the probability the kth word will occur is $\hat{\theta}_k = F_k/n$.

Typically, the maximum likelihood estimator is not used because Laplace smoothing is often introduced (a standard practice when using multinomials for problems like text mining). With Laplace smoothing, the estimator becomes

$$\tilde{\theta}_k = \frac{F_k + 1}{\sum_{k'=1}^{d}(F_{k'} + 1)} = \frac{n\hat{\theta}_k + 1}{n + d}. \tag{8.13}$$

Note that we will use the notation $\hat{\theta}$ to describe parameter estimates when Laplace smoothing has not been used and $\tilde{\theta}$ to indicate parameter estimates when Laplace smoothing has been used.

After random oversampling, we find that the parameter estimates learned from the resampled points and original training set if Laplace smoothing is used are:

$$E\left[\tilde{\theta}_k^{(r)}\right] = \frac{(n+m)\theta_k + 1}{n + m + d} \tag{8.14}$$

and

$$\text{Var}\left[\tilde{\theta}_k^{(r)}\right] = \left[1 + \frac{m(n-2d-1) - d(2n+d)}{(n+m+d)^2}\right] \frac{\theta_k(1-\theta_k)}{n}. \tag{8.15}$$

Thus, provided $m < d(2n+d)/(n-2d-1)$, the variance of the pooled, smoothed estimates will be smaller than that of the original sample.

In generative oversampling, one generates points based on an assumed probability distribution. If one uses a multinomial model to generate the points, the parameters one uses in the initial estimation can be either $\hat{\theta}$ (i.e., without Laplace smoothing) or $\tilde{\theta}$ (i.e., with Laplace smoothing). When using generative oversampling with multinomial naive Bayes, there are two places where Laplace smoothing can possibly be used: when performing the initial parameter estimates for generative oversampling and when performing the parameter estimates for multinomial naive Bayes. In our experiments, we always use Laplace smoothing when estimating parameters for multinomial naive Bayes. The question of whether to use Laplace smoothing will therefore always refer to the initial parameter estimates in generative oversampling.

As shown in the Appendix, if one uses $\hat{\theta}$ for initial parameter estimates in generative oversampling, then $E\left[\tilde{\theta}_k^{(g)}\right] = E\left[\tilde{\theta}_k^{(r)}\right]$ and $\text{Var}\left[\tilde{\theta}_k^{(g)}\right] = \text{Var}\left[\tilde{\theta}_k^{(r)}\right]$. If one performs Laplace smoothing and uses $\tilde{\theta}$ in the initial parameter estimates used for generative oversampling, however, the parameter vector $\tilde{\theta}^{(g)}$ estimated after generative oversampling will be different from the parameter vector $\tilde{\theta}^{(r)}$ estimated after random oversampling or the parameter vector $\tilde{\theta}$ used in cost-sensitive learning. Our empirical results show that the relative performance of using either $\hat{\theta}$ or $\tilde{\theta}$ when estimating the initial parameters used in generative oversampling depends on which classifier is used.

Since a cost-sensitive naive Bayes classifier uses the parameter estimates $\tilde{\theta}$ from the original set of points, it is clear that there will be a difference, in expectation, between the parameters estimated via random oversampling, generative oversampling, and cost-sensitive learning. We will see empirically that resampling produces

better classifiers than cost-sensitive learning. Thus, even though resampling incurs additional overhead in terms of time and memory, the improvement in classification may justify this additional effort.

8.6 Empirical Comparison of Resampling and Cost-Sensitive Learning

In this section, we will provide empirical support for our analysis in Section 8.5. We will show that, as predicted, there is minimal empirical difference between random oversampling, generative oversampling, and cost-sensitive learning when Gaussian naive Bayes is used as the classifier. In contrast, when dealing with high-dimensional text data sets where a multinomial model is more suitable, there is a difference between random oversampling, generative oversampling, and cost-sensitive learning. The magnitude of the difference with regard to generative oversampling is related to whether Laplace smoothing is used to build the model used to generate artificial points.

While the primary goal of this chapter is not to perform extensive empirical benchmarks of all possible resampling methods, for the sake of comparison, we have also included some common resampling techniques (namely SMOTE and random undersampling).

8.6.1 Explaining Empirical Differences Between Resampling and Cost-Sensitive Learning

Our experiments compare the results of classifiers learned after resampling against a cost-sensitive classifier that estimates its parameters from the original training set. In this section, we will describe why comparing naive Bayes after resampling with cost-sensitive naive Bayes can answer the question of whether the benefits of resampling are limited to merely evening out the imbalance of the class priors, or if additional effects (from changing the estimates of the likelihoods) are responsible.

Oversampling the positive class has two possible effects on a Bayesian classifier: (1) it changes the estimated priors $\hat{P}(Y = y_+)$ and $\hat{P}(Y = y_-)$ to $\hat{P}^{(rs)}(Y = y_+)$ and $\hat{P}^{(rs)}(Y = y_-)$ and (2) it may or may not change the parameter estimate $\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$ such that $\hat{P}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+) \neq \hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})$.

The decision rule of the Bayesian classifier after resampling is to assign a point x to the positive class if

$$\frac{\hat{P}^{(rs)}(Y = y_+|\mathbf{X} = \mathbf{x})}{\hat{P}^{(rs)}(Y = y_-|\mathbf{X} = \mathbf{x})} = \frac{\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_+, \hat{\theta}_+^{(rs)})\hat{P}^{(rs)}(Y = y_+)}{\hat{P}^{(rs)}(\mathbf{X} = \mathbf{x}|Y = y_-, \hat{\theta}_-^{(rs)})\hat{P}^{(rs)}(Y = y_-)} \geq 1. \quad (8.16)$$

As described in the previous section, if $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+)=\hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+^{(rs)})$ and $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_-,\hat{\theta}_-)=\hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_-,\hat{\theta}_-^{(rs)})$, then the only effect of resampling is to change the learned class priors. Under this (possibly incorrect) assumption,

$$\frac{\hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+^{(rs)})\hat{P}^{(rs)}(Y=y_+)}{\hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_-,\hat{\theta}_-^{(rs)})\hat{P}^{(rs)}(Y=y_-)}=\frac{\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+)\hat{P}^{(rs)}(Y=y_+)}{\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_-,\hat{\theta}_-)\hat{P}^{(rs)}(Y=y_-)}. \tag{8.17}$$

This is the same as adjusting the decision rule learned on the original training set by setting $\alpha=\frac{\hat{P}^{(rs)}(Y=y_-)\hat{P}(Y=y_+)}{\hat{P}^{(rs)}(Y=y_+)\hat{P}(Y=y_-)}$ and assigning $\mathbf{x}$ to the positive class if $\frac{\hat{P}(Y=y_+|\mathbf{X}=\mathbf{x})}{\hat{P}(Y=y_-|\mathbf{X}=\mathbf{x})}\geq\alpha$. One can do this by training a cost-sensitive naive Bayes classifier with $\alpha=\frac{c_-}{c_+}=\frac{\hat{P}^{(rs)}(Y=y_-)\hat{P}(Y=y_+)}{\hat{P}^{(rs)}(Y=y_+)\hat{P}(Y=y_-)}$.

Thus, one can duplicate the beneficial effect of evening out the class priors via resampling if $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+)=\hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+^{(rs)})$ and $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_-,\hat{\theta}_-)=\hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_-,\hat{\theta}_-^{(rs)})$ by using a cost-sensitive naive Bayes classifier where $\frac{c_-}{c_+}=\frac{\hat{P}^{(rs)}(Y=y_-)\hat{P}(Y=y_+)}{\hat{P}^{(rs)}(Y=y_+)\hat{P}(Y=y_-)}$. Any empirical difference observed between a naive Bayes classifier after resampling and a cost-sensitive naive Bayes classifier with the appropriate values of c_- and c_+ is therefore attributable to the fact it is incorrect to assume that the estimated parameters modeling our probability distributions are equal before and after resampling.

Therefore, we can examine whether $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+)=\hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+^{(rs)})$ by comparing a naive Bayes classifier that uses resampling and an equivalent cost-sensitive naive Bayes classifier where $\frac{c_-}{c_+}=\frac{\hat{P}^{(rs)}(Y=y_-)\hat{P}(Y=y_+)}{\hat{P}^{(rs)}(Y=y_+)\hat{P}(Y=y_-)}$. Such a comparison allows us to isolate and study only the part of resampling that could cause $\hat{P}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+)\neq\hat{P}^{(rs)}(\mathbf{X}=\mathbf{x}|Y=y_+,\hat{\theta}_+^{(rs)})$. We perform this comparison in Sections 8.6.2 and 8.6.3.

8.6.2 Naive Bayes Comparisons on Low-Dimensional Gaussian Data

In this section, we will provide some simple examples of classifying low-dimensional data with a Gaussian naive Bayes classifier to support the theory presented in Section 8.5.3.

We use f1-measure, a natural evaluation metric in information retrieval for high-dimensional data sets, as our evaluation metric. F1-measure is the harmonic mean of two other evaluation metrics, precision and recall. Precision = $n_{tp}/(n_{tp}+n_{fp})$ and recall = $n_{tp}/(n_{tp}+n_{fn})$, where n_{tp} is the number of true positives, n_{fp} is the number of false positives, and n_{fn} is the number of false negatives. F1-measure

ranges from 0 to 1, with 1 being the best possible f1-measure achievable on the test set. F1-measure has some additional advantages over traditional ROC and AUC metrics when interpreting our experiments, as discussed in Section 8.6.2.2. In order to keep our results consistent, we use f1-measure for both the low-dimensional and the high-dimensional experiments.

To illustrate that there is minimal benefit in using either random oversampling, generative oversampling, or cost-sensitive learning when a Gaussian naive Bayes classifier is used, we present two sets of experiments.

8.6.2.1 Gaussian Naive Bayes on Artificial, Low-Dimensional Data

The first set of experiments utilizes an artificially generated data set consisting of two classes drawn from two one-dimensional Gaussians with true variance equal to 1. The location of the means of the two Gaussians is controlled such that a specific optimal Bayes error rate could potentially be achieved if the points in the test data were sampled equally from the two distributions (the Bayes error rate is defined as the lowest theoretical error rate achievable if we knew the true values of the parameters in our mixture of Gaussians [5]). We vary the optimal Bayes error rate (denoted as BER in Fig. 8.1) between 0.1 and 0.3. In order to introduce imbalance, 90% of the training set consists of points in the negative class and 10% of the training set

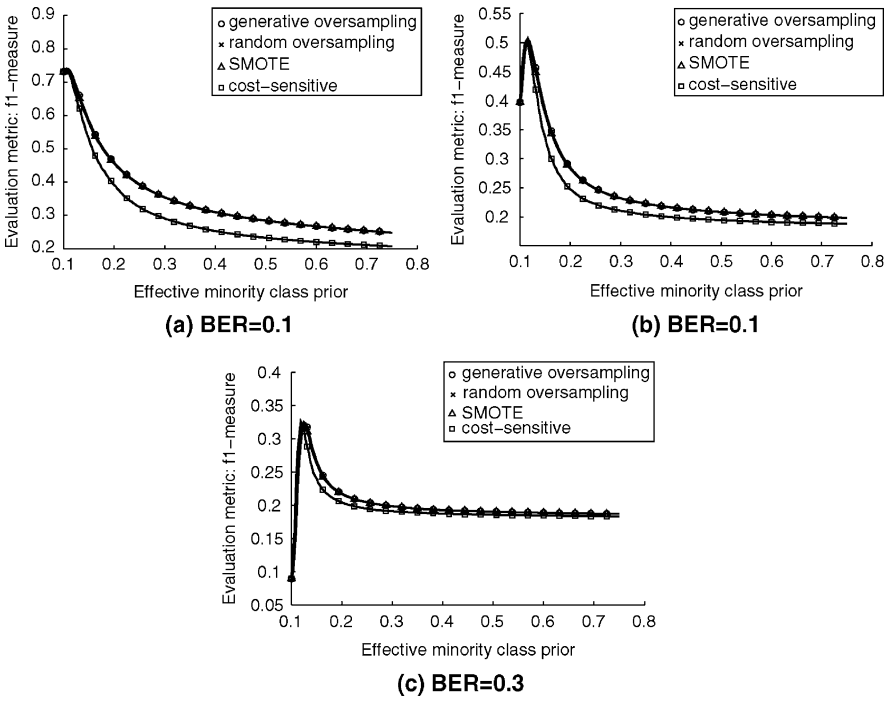


Fig. 8.1 Results on artificial Gaussian data for naive Bayes

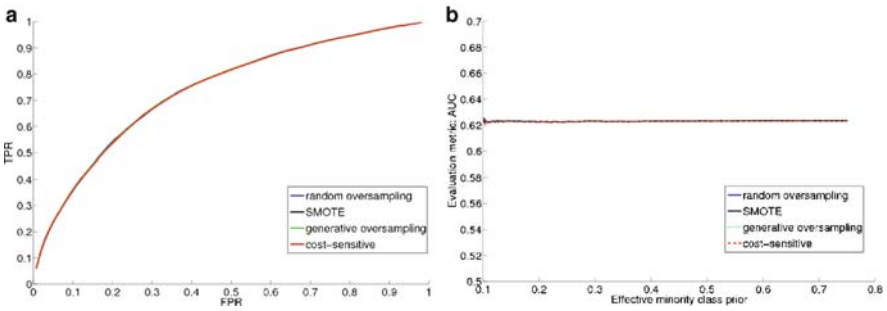
consists of points in the positive class. In our experiments, we varied the amount of training data between 100 and 300 points, but found that the results were consistent regardless of how much training data was used; here, we present results where there are 100 training points. The test set consists of 1000 points with the same priors as the training set. We average our results over 100 trials, where each trial includes creating a completely new data set.

When resampling, one has control over the value of the estimated priors $\hat{P}^{(rs)}(Y = y_+)$ and $\hat{P}^{(rs)}(Y = y_-)$. Since $\hat{P}^{(rs)}(Y = y_+) + \hat{P}^{(rs)}(Y = y_-) = 1$, controlling $\hat{P}^{(rs)}(Y = y_+)$ is sufficient to control both $\hat{P}^{(rs)}(Y = y_+)$ and $\hat{P}^{(rs)}(Y = y_-)$. In our experiments, we vary $\hat{P}^{(rs)}(Y = y_+)$ between the prior estimated without resampling $\hat{P}(Y = y_+) = 10\%$ and a maximum possible value of 75%. Note that when $\hat{P}^{(rs)}(Y = y_+) = \hat{P}(Y = y_+)$, no actual resampling has been performed (this corresponds to the left-most point on each graph plotting f1-measure where all performance curves converge). When running cost-sensitive learning, we control the misclassification costs c_- and c_+. In order to directly compare cost-sensitive learning against results for resampling, we use the term "effective minority class prior" in our graphs. That is, a particular value of the effective minority class prior means that (1) when resampling is used, the resampled prior $\hat{P}^{(rs)}(Y = y_+)$ is equal to the effective minority class prior and (2) when cost-sensitive learning is used, $\frac{c_-}{c_+} = \frac{\hat{P}^{(rs)}(Y=y_-)\hat{P}(Y=y_+)}{\hat{P}^{(rs)}(Y=y_+)\hat{P}(Y=y_-)}$, where $\hat{P}^{(rs)}(Y = y_+)$ is equal to the effective minority class prior. In interpreting our results, we simply look at our results on the test set across a range of resampling rates. Choosing a resampling rate that yields optimal performance is an unsolved problem. That is, there is no closed-form solution for determining the appropriate effective minority class prior to maximize a particular evaluation metric, so this becomes a model selection problem.

In Fig. 8.1, we plot the f1-measure versus different effective minority class priors for Gaussian naive Bayes after random oversampling, Gaussian naive Bayes after generative oversampling, Gaussian naive Bayes after SMOTE, and cost-sensitive naive Bayes. Interestingly, regardless of the separability of the two Gaussian distributions, the curves have similar characteristics. In particular, the best possible f1-measure obtained by each is almost exactly the same. That is, in practice, random oversampling, generative oversampling, and SMOTE seem to have little effect on Gaussian naive Bayes that cannot be accomplished via cost-sensitive learning. This supports the theory presented in Section 8.5.3, which shows that, in expectation, the value of the sample mean estimated after random oversampling, generative oversampling, or cost-sensitive learning (which uses the original set of points) is equivalent.

8.6.2.2 A Note on ROC and AUC

Figure 8.2 contains ROC curves and values for AUC for the same set of experiments presented in Fig. 8.1 where BER = 0.3. When evaluating results on imbalanced data sets, ROC and AUC are often very useful. However, ROC and AUC can hide the



(a) Example ROC curves for artificial data (BER = 0.3)

(b) Example AUC curves for artificial data (BER = 0.3)

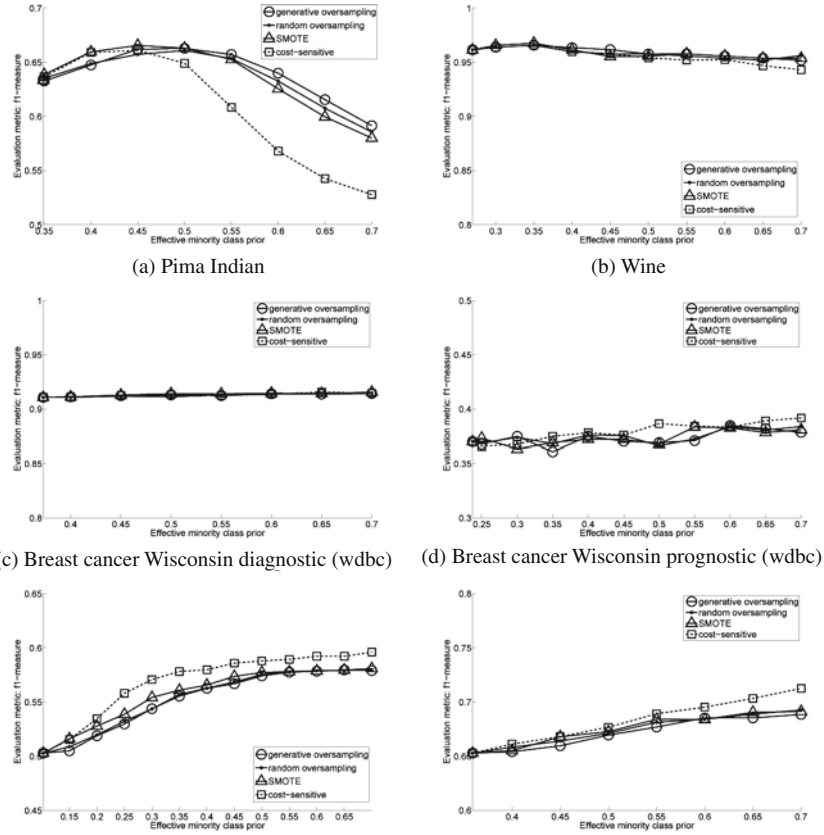
Fig. 8.2 Example results using AUC and ROC as evaluation metrics; note that using these evaluation metrics, it is difficult to determine whether the effective minority class prior has any effect on how well the classifier performs

effect that different resampling rates have on the classifier. To fully examine the effect of resampling rate using ROC curves would require an unwieldy number of curves per graph, since each resampling rate for each method being used would require a separate ROC curve. Figure 8.2, which seems to contain only a single ROC curve, is an exception since, as theory predicts, each ROC curve produced by each resampling method and cost-sensitive Gaussian naive Bayes is essentially the same (thus retraced multiple times in the figure). Another problem is that to create a ROC curve, one uses several different thresholds for each point on the curve; cost-sensitive naive Bayes also uses different thresholds to produce results using different costs. Thus, the differences in performance across different costs are hidden on a single ROC curve for this type of classifier. AUC, which is based on ROC, aggregates results over several thresholds. Thus, the AUC for cost-sensitive naive Bayes will always be about the same (sans statistical variation in the training/test sets) regardless of the cost used and is not particularly interesting. In fact, this is exactly what we see in Fig. 8.2, where the AUC remains essentially constant regardless of the effective minority class prior. The results in Fig. 8.2 can be extremely misleading, because it may lead one to conclude that different cost parameters or resampling rates have no effect on how well a classifier performs. In comparison, using f1-measure to plot unintegrated results corresponding to specific resampling rates and costs clearly shows the importance of choosing an appropriate resampling rate or cost.

8.6.2.3 Gaussian Naive Bayes on Real, Low-Dimensional Data

To complement the experiments on Gaussian naive Bayes on artificial data, we also present some results on low-dimensional data sets from the UCI data set repository

to verify generalizations of the findings on real data. We have selected six data sets: pima indian, wine, breast cancer Wisconsin diagnostic (wdbc), breast cancer Wisconsin prognostic (wpbc),[1] page-blocks (using "non-text" versus "rest" as our binary class problem), and ionosphere. The features of all of the dimensions for wine and breast cancer Wisconsin diagnostic (wdbc) pass the Kolmogorov–Smirnov test for normality for a p value of 0.05; most of the features of the breast cancer Wisconsin prognostic (wpbc) data set also pass the Kolmogorov–Smirnov test for normality. In contrast, the majority of the features of the remaining data sets did not. We use both features that passed and did not pass the Kolmogorov–Smirnov test in our experiments, so the assumption that Gaussians can be used to model the various data sets does not hold very well in some cases.



(a) Pima Indian (b) Wine

(c) Breast cancer Wisconsin diagnostic (wdbc) (d) Breast cancer Wisconsin prognostic (wdbc)

(e) Page blocks (non-text vs. rest) (f) Ionosphere

Fig. 8.3 Results on real data sets from UCI data set repository using Gaussian naive Bayes

[1] Note that the two breast cancer data sets are separate data sets in the UCI data set repository and not the same data set used for two different tasks in our experiments.

Our results are shown in Fig. 8.3. The results of these experiments support the same conclusion as before: there is little advantage to using either random oversampling, generative oversampling, or cost-sensitive learning when using Gaussian naive Bayes. For the sake of comparison, SMOTE is included in these datasets as well. While SMOTE has been shown to work well with other classifiers [4], it performs similarly to the other techniques when using Gaussian naive Bayes. Thus, given that it is much easier to use cost-sensitive learning instead of resampling and that there is no empirical advantage of using resampling, it is preferable to simply use a cost-sensitive version of naive Bayes when Gaussian distributions are used to model the data.

8.6.3 Multinomial Naive Bayes

In this section and the next, we examine the empirical effect of resampling on high-dimensional data. In particular, we will use text classification as an example domain. We first examine the effect of random and generative oversampling on multinomial naive Bayes [17], a classifier often used for text classification. Our experiments on text classification are more extensive than the experiments on low-dimensional data for two primary reasons: (1) additional results more fully illustrate the empirical differences between the different resampling methods and (2) empirical studies comparing resampling methods and/or cost-sensitive learning typically focus on low-dimensional data, so there are less published results available for high-dimensional data.

We compare the effect of random oversampling, generative oversampling, and cost-sensitive learning on multinomial naive Bayes using six text data sets drawn from different sources. The text data sets come from several past studies on information retrieval including TREC (http://trec.nist.gov), OHSUMED [8], and WebAce [7]. The data sets from TREC are all newspaper stories from either the LA Times (la12 data set) or the San Jose Mercury (hitech, reviews, sports data sets) classified into different topics. The ohscal data set contains text related to medicine, while the k1b data set contains documents from the Yahoo! subject hierarchy. All six of these data sets were included in the CLUTO toolkit [11].[2]

All text data sets 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[3] (see [22] for a discussion of TFIDF weighting and other common preprocessing tasks in text classification). Finally, we created several two-class problems based on our text data sets. For each data set, we chose the smallest class

[2] We use the data sets available at http://www.ideal.ece.utexas.edu/data/docdata.tar.gz, which have some additional preprocessing as described in [28].

[3] Note that the order in which one applies TFIDF weighting, normalization, and resampling appears to be important in terms of generating good classification performance; further analysis is required to determine the reasons for this.

in each data set as the minority class and aggregated all other classes to form the majority class.

For each data set, 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 again use f1-measure as our evaluation metric and results are averaged over the ten training/test splits.

As in the experiments with Gaussian data, we control the effective minority class prior either through resampling or through cost-sensitive learning. Again, we vary the effective minority class prior between the prior estimated without resampling $\hat{P}(Y = y_+)$ and 70% (note that the prior before resampling varies for each data set but is always less than 10%).

Details about these data sets are given in Table 8.1, including the number of minority class points in the data and the "natural" value of the minority class prior in the data set.

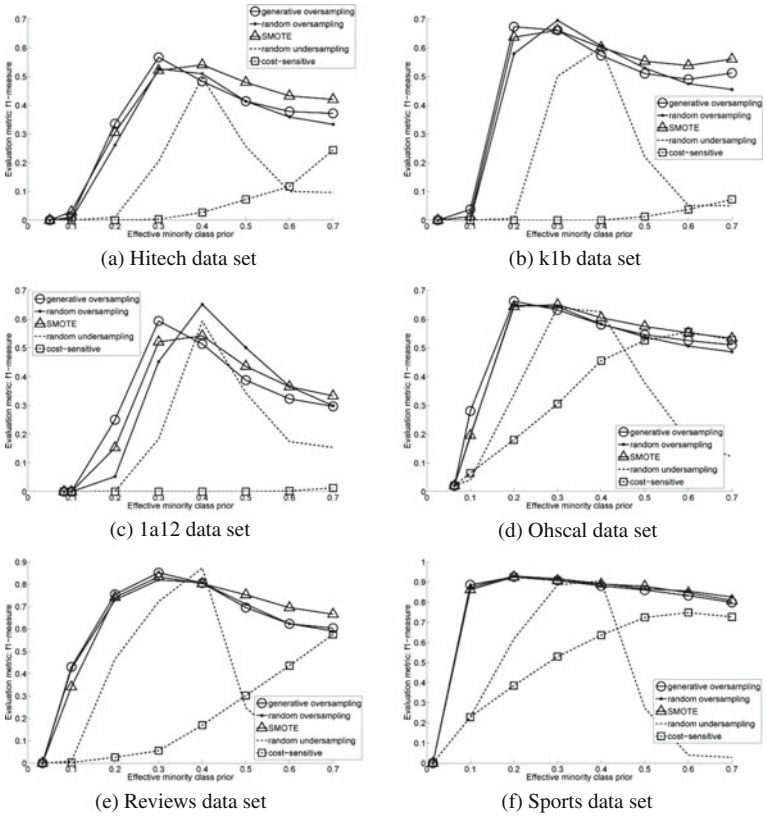
Table 8.1 Data set characteristics

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

The results of our experiments on multinomial naive Bayes are shown in Fig. 8.4. The results indicate that resampling improves the resulting f1-measure when compared to classification without resampling (i.e., the left-most point in each graph). The results also indicate that there is a significant difference between the best possible f1-measure obtained from resampling (across all resampling rates) and the best possible f1-measure obtained through cost-sensitive learning (across all tested costs). In particular, the best possible f1-measure obtained after oversampling (regardless of which oversampling method we tested) is always better than cost-sensitive learning. In some cases, this value is much higher than the best possible f1-measure obtained via cost-sensitive learning.

Random oversampling, generative oversampling,[4] and SMOTE produce comparable f1-measure curves. These results indicate that, in practice, one can produce a classifier with better performance by oversampling instead of adjusting the decision boundary using cost-sensitive learning.

[4] Here, we use generative oversampling as described in the Appendix where $\hat{\theta}$ (i.e., without smoothing during parameter estimation) is used instead of $\tilde{\theta}$ to generate points; we will see in Section 8.6.4 why this distinction is important.



(a) Hitech data set (b) k1b data set

(c) 1a12 data set (d) Ohscal data set

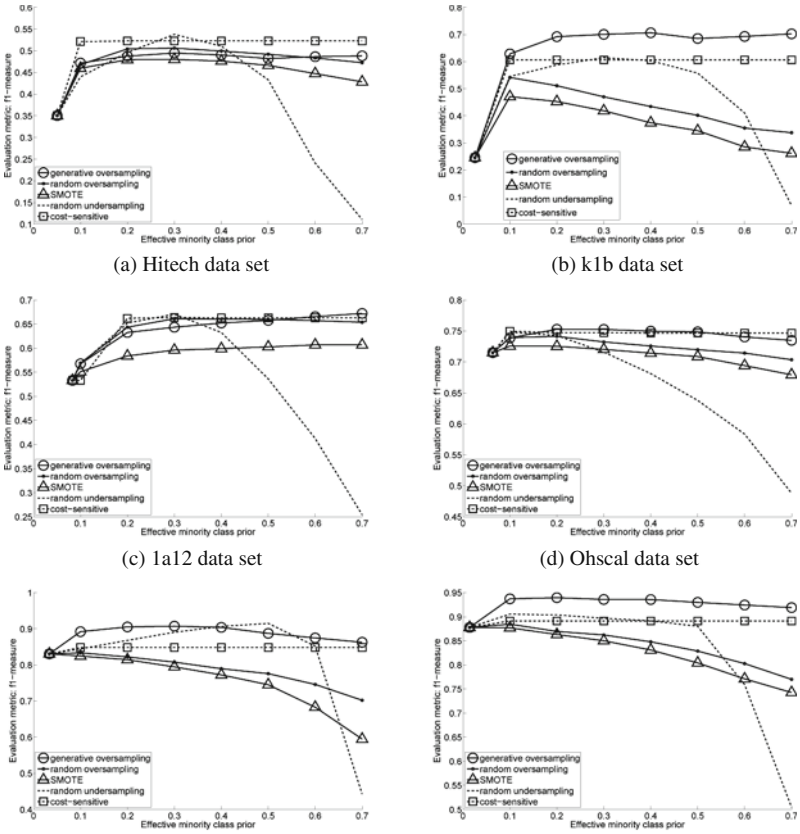
(e) Reviews data set (f) Sports data set

Fig. 8.4 Results on text data sets using multinomial naive Bayes

8.6.4 SVMs

In this section, we empirically test the effect of resampling on linear SVMs in the domain of text classification and compare performance against cost-sensitive SVMs.[5] In our experiments, we use an SVM with a linear kernel, which has been shown to work well in text classification [25]. Specifically, we use the SVM light implementation by Joachims [10]. 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

[5] All SVM results presented use generative oversampling for multinomials with smoothing during parameter estimation ($\tilde{\theta}$) except for Fig. 8.7, where we present results for generative oversampling with both $\tilde{\theta}$ and $\hat{\theta}$. Generative oversampling for multinomials without smoothing (i.e., when $\hat{\theta}$ is used when estimating the parameters for generative oversampling) performs poorly for SVMs as shown at the end of this section.



(a) Hitech data set (b) k1b data set

(c) 1a12 data set (d) Ohscal data set

(e) Reviews data set (f) Sports data set

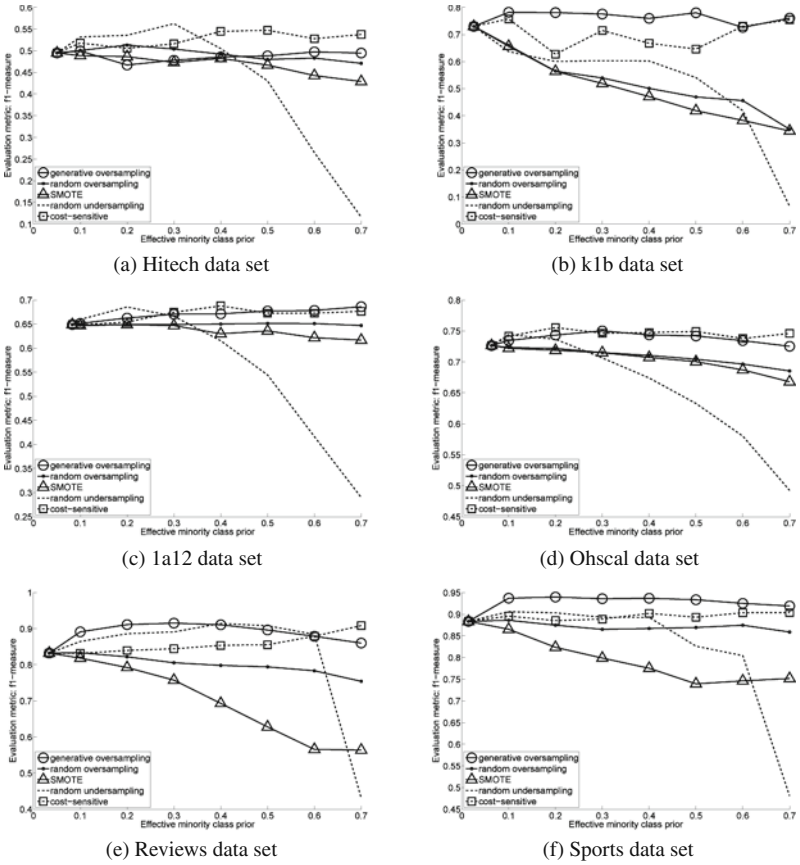
Fig. 8.5 Results on text data sets using SVMs without tuning C

the parameter C.[6] 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 and the results are presented separately in Fig. 8.5 and Fig. 8.6.

Figure 8.5 plots our results comparing generative oversampling, random oversampling, SMOTE, random undersampling, and cost-sensitive SVMs on each of the six data sets when all default parameters for linear SVMs are used. The plots show f1-measure as a function of effective minority class prior, as presented for the naive Bayes results.

Figure 8.6 shows the results when the trade-off C between margin size and number of training errors is tuned via a validation set. 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 the initial training set) and validation set (the

[6] Not to be confused with a cost matrix **C**.



(a) Hitech data set (b) k1b data set

(c) 1a12 data set (d) Ohscal data set

(e) Reviews data set (f) Sports data set

Fig. 8.6 Results on text data sets using SVMs while tuning C

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 data set, training set split, resampled minority class prior, and resampling method).

In both sets of experiments, generative oversampling performs well compared to random oversampling, SMOTE, random undersampling, and cost-sensitive SVMs. Generative resampling also shows robustness to the choice of the minority class prior (i.e., its performance does not vary significantly across resampling rates), which may be otherwise difficult to optimize, in practice. 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 minority class training prior. In all six of our data sets, there is a very clear degradation in f1-measure for random undersampling when the minority class training prior is either too low or too high regardless of whether C is tuned. Cost-sensitive SVMs perform quite well and are as robust to choice of cost parameter as generative oversampling is to choosing how much to resample, but on average, generative oversampling produces a higher f1-measure.

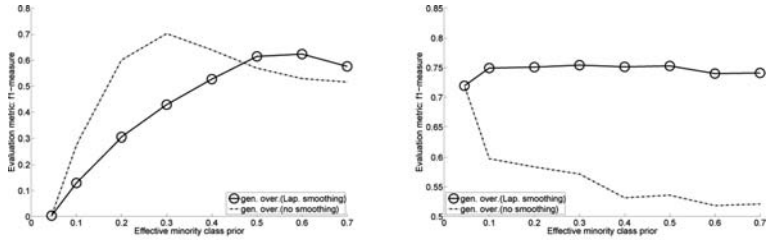


Fig. 8.7 Average results on text data sets for multinomial naive Bayes and SVMs (where C is tuned) after running generative oversampling with different levels of smoothing

If one runs SVMs with resampled points created via generative oversampling with Laplace smoothing during parameter estimation (i.e., if one uses $\tilde{\theta}$), then 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. Figure 8.7 contains averaged results where we compare generative oversampling on SVMs using either $\tilde{\theta}$ or $\hat{\theta}$ (i.e., generative oversampling with and without Laplace smoothing). As one can see, the results when using $\hat{\theta}$ are much worse. When $\hat{\theta}$ is used, generative oversampling no longer effectively creates points outside of the original convex hull. Thus, generative oversampling with $\tilde{\theta}$ complements the SVMs well by increasing the size of the minority class convex hull.

Note that, in our multinomial naive Bayes experiments, we found that using $\hat{\theta}$ was always empirically superior to using $\tilde{\theta}$, while for SVMs, we found that using $\tilde{\theta}$ resulted in better empirical performance (see Fig. 8.7 for graphs of each case). That is, even the same resampling method in the same problem domain interacts very differently with different classifiers.

Finally, we observe that all of the results that work well with SVMs move the separating hyperplane in some fashion, either by (1) changing the shape of the convex hulls inscribing either the minority class (generative oversampling) or the majority class (random undersampling) or (2) changing the location of the separating hyperplane by controlling the trade-off between margin and number of empirical mistakes in the training set (tuning the parameter C or using cost-sensitive SVMs). Our analysis supports the conclusion that random oversampling and SMOTE, which work well for the multinomial naive Bayes classifier, have minimal effect on SVMs since neither are effective at changing the shape of the minority class convex hull. In fact, if one tunes the parameter C, then neither random oversampling nor SMOTE is useful.

8.6.5 Discussion

In summary, we have presented experiments which can be divided into two cases: in the first case, there is no advantage to performing resampling as opposed to simply performing cost-sensitive learning. Examples of this were presented for artificial and real low-dimensional data sets for a Gaussian naive Bayes classifier. In

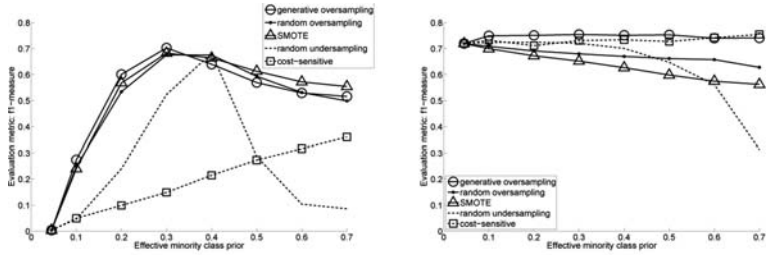


Fig. 8.8 Average results on text data sets for multinomial naive Bayes and SVMs (where C is tuned)

the second case, there is a clear difference in performing resampling as opposed to cost-sensitive learning due to various effects caused by the resampling methods. Several examples of this effect were presented using multinomial naive Bayes and linear SVMs on high-dimensional text data sets. Empirically, we showed that, for both naive Bayes and SVMs on text classification, resampling resulted in a better classifier than cost-sensitive learning.

Averaged results for the text data sets used in this chapter are presented in Fig. 8.8. When trying to achieve the optimal f1-measure on these text data sets, Fig. 8.8 shows that the best approach is to use linear SVMs with generative oversampling.

Our results support the conclusion that the best resampling method (or whether cost-sensitive learning outperforms resampling) is dependent on the data set and classifier being used. This has been seen empirically in other papers such as [9]. Our analysis provides some insight for why these differences occur.

8.7 Conclusion

In this chapter, we have analyzed the relationship between resampling and cost-sensitive learning. In particular, we examine the effect of random and generative oversampling versus cost-sensitive learning from a theoretical perspective using Bayesian classifiers. The theoretical analysis is supported by empirical results, where we briefly examined the effect of resampling on low-dimensional data and included a much more extensive treatment of resampling versus cost-sensitive learning on high-dimensional text data sets using multinomial naive Bayes and linear SVMs.

Results vary depending on the dataset and the classifier used. In particular, for low-dimensional data sets where a Gaussian distribution is appropriate to model the classes, there seems to be no advantage to using resampling. Theoretically, resampling results in the same expected sample mean but with greater variance. Empirically, there is no benefit to resampling over cost-sensitive learning when used

with a Gaussian naive Bayes classifier. In practice, this means that resampling is unnecessary if Gaussian naive Bayes is used; a cost-sensitive classifier that performs just as well can easily be trained without the overhead of resampling. When applying multinomial naive Bayes to text data sets, resampling results in changed class priors as well as different estimates of the parameters of the multinomial distribution modeling the resampled class. In this case, any of the oversampling methods tested result empirically in better classification of the minority class. Finally, when classifying imbalanced text data sets using an SVM classifier, we see that using generative oversampling, which helps to expand the convex hull of the minority class, can lead to consistently good performance. In particular, the best overall performance when classifying text data sets, regardless of classifier or method of resampling/incorporating costs, is generative oversampling coupled with SVMs.

Two of the most important results described in this chapter are as follows. First, while there is a theoretical equivalence between cost-sensitive learning and resampling under certain assumptions, these assumptions are often broken in practice. For example, all of the experiments on high-dimensional text data sets (for both naive Bayes and SVMs) break these assumptions, leading to an observed empirical difference between cost-sensitive and resampling methods. We also show that there is no resampling method that is always best. We give analytical and empirical results supporting why different resampling methods interact differently with certain classifiers on certain types of data.

Both of these results help explain why there are often differences between cost-sensitive learning and resampling methods in empirical studies such as [24]. Several areas of future work remain to explore other differences and further explain effects observed herein.

Appendix

In this section, we will derive some of the equations found in this chapter, particularly those in Sections 8.5.3 and 8.5.4.

Gaussian Random Oversampling

Let the set of n points in the positive class be denoted as $X_1, \ldots, X_n$. These points are a random sample from a normal distribution with true mean μ and true variance σ^2. Let us now consider a random sample $X_1^{(r)}, \ldots, X_m^{(r)}$ produced through random oversampling. Thus, each $X_j^{(r)}$ equals some X_{K_j}, where $K_1, \ldots, K_m$ are i.i.d. uniformly distributed random variables over the discrete set $\{1, \ldots, n\}$. The sample mean, $\bar{X}^{(r)}$, of only the randomly resampled data is an unbiased estimator of the true mean, since