Classification on Data with Biased Class Distribution

Slobodan Vucetic[1] and Zoran Obradovic[2]

[1] School of Electrical Engineering and Computer Science, Washington State University,
Pullman, 99164 WA, USA
svucetic@eecs.wsu.edu
[2] Center for Information Science and Technology, Temple University,
Philadelphia, PA 19122, USA
zoran@ist.temple.edu

Abstract. Labeled data for classification could often be obtained by sampling that restricts or favors choice of certain classes. A classifier trained using such data will be biased, resulting in wrong inference and sub-optimal classification on new data. Given an unlabeled new data set we propose a bootstrap method to estimate its class probabilities by using an estimate of the classifier's accuracy on training data and an estimate of probabilities of classifier's predictions on new data. Then, we propose two methods to improve classification accuracy on new data. The first method can be applied only if a classifier was designed to predict posterior class probabilities where predictions of an existing classifier are adjusted according to the estimated class probabilities of new data. The second method can be applied to an arbitrary classification algorithm, but it requires retraining on the properly resampled data. The proposed bootstrap algorithm was validated through experiments with 500 replicates calculated on 1,000 realizations for each of 16 choices of data set size, number of classes, prior class probabilities and conditional probabilities describing a classifier's performance. Applications of the proposed methodology to a benchmark data set with various class probabilities on unlabeled data and balanced class probabilities on the training data provided strong evidence that the proposed methodology can be successfully used to significantly improve classification on unlabeled data.

1 Introduction

A common assumption made in machine learning is that labeled data used for training a classifier and unlabeled new data can be considered as samples from the same underlying distribution. In such a case one could apply standard machine learning procedures to learn a classifier from labeled data (e.g., logistic regression, decision trees, neural networks), estimate its accuracy (e.g., directly from training set, using cross-validation), and apply it on unlabeled examples in a straightforward manner. However, this assumption is often violated with labeled and/or unlabeled data obtained by biased sampling from an underlying distribution. While inference and learning in such a general setup is an open machine learning problem, in this paper we propose a methodology for solving an important special case where class distributions

in labeled and/or unlabeled data are biased. To simplify the presentation it will be assumed that unlabeled data is a sample from an underlying distribution. This corresponds to the goal of constructing a classifier optimized for successful predictions on an unlabeled data set.

A familiar example that involves biased class distribution is classification of a rare medical condition where false negative predictions can have high costs. A common approach is to intentionally provide a biased training set with a disproportionally large number of examples from the rare condition in order to produce a successful classifier with a small fraction of false negative predictions. Another example is the case where costs of obtaining labeled examples are class-dependent and where the resulting labeled data set is biased towards examples from less expensive classes.

An interesting example that motivated this work is the problem of predicting protein disorder from its amino-acid sequence [11,13]. Protein disorder is a biological concept that refers to proteins that do not crystallize into a unique 3D structure [7]. To obtain training set for prediction of protein disorder one should collect representative examples of ordered and disordered proteins. However, since protein disorder is an insufficiently explored phenomenon, accurate estimates of its commonness in nature do not exist. Additionally, current databases of proteins with known structure are highly biased towards ordered proteins. The explanation is that crystallographers are reluctant to publish structural results for disordered proteins since there is always a risk that some procedural error prevented proper protein crystallization. The proposed methodology could lead to a successful predictor of protein disorder and provide confident estimates of protein disorder commonness.

The learning problem considered in this paper can be defined as construction of a classifier using labeled data of size n_L with class probabilities $P_L(i)$, $i = 1, \ldots c$, where c is the number of classes, for accurate prediction on unlabeled data of size n_U with unknown class probabilities $P_U(i)$. This problem can be solved by (*step*1) estimating the class distribution $P_U(i)$, followed by (*step*2) using this estimate to construct a desired classifier. We propose a bootstrap methodology [8] to estimate distribution of $P_U(i)$ based on (*step*1.*a*) an estimate of classifier's accuracy obtained on labeled data, and (*step*1.*b*) an estimate of classifier's class predictions on unlabeled data.

To construct a successful classifier (*step*2) we propose two approaches depending on the type of classification algorithm and available computational resources. If a classifier was trained to estimate posterior class probabilities (e.g., logistic regression, neural networks) it is possible to use the existing classifier on an unlabeled data set by adjusting its outputs according to estimated $P_U(i)$. If computational resources allow, or if a classifier represents a nonlinear discriminant function that directly provides classification (e.g., decision trees) we propose a procedure for retraining of classifier using resampled labeled data according to estimated $P_U(i)$.

In real-life applications of classification different costs are often associated with different types of errors. Misclassification costs are usually described by cost matrix C with elements $C(j, i)$ representing the costs of predicting class j when the true class is i. Unless explicitly mentioning otherwise, in this paper we assume 0/1 loss, where $C(j, i) = 1$ if $i \neq j$ and $C(i, i) = 0$, to simplify presentation. However, it is important to note that the proposed methodology is not restricted to the choice of cost matrix so that it can be generalized to an arbitrary cost matrix.

2 Estimation of the Class Distribution on an Unlabeled Data Set

Let us assume that a labeled data set S_L with n_L examples is available to learn a classifier and to estimate its accuracy, while the constructed classifier should be applied to a new unlabeled data set S_U with n_U examples. Classification accuracy is completely determined by (*i*) conditional probabilities $p(j|i)$ of predicting a class j if the true class is i, and (*ii*) prior probabilities p_i (shorthand for $P_U(i)$) of class i on S_U, where $i, j = \{1, ..., c\}$, and c represents the number of classes. The 0/1 loss or error rate can then be calculated as

$$error\_rate = \sum_j \sum_i p(j|i) \cdot p_i \cdot (1-\delta_{ji}) \cdot 100[\%], \tag{1}$$

where δ_{ij} is the Kronecker's delta with $\delta_{ij} = 1$ for $i = j$ and zero otherwise.

Calculating Class Probabilities from Very Large Data Sets

Class probabilities p_i on S_U are not known in advance, so they should be estimated using an available classifier with known conditional probabilities $p(j|i)$. Note that the only information that could be obtained using the classifier on unlabeled data S_U are its predictions. From these predictions we can estimate the probability of predicting class j on S_U denoted as q_j. The connection between p_i and q_j can be expressed as

$$q_j = \sum_i p(j|i) p_i, \tag{2}$$

or as $\mathbf{q} = \mathbf{P} \cdot \mathbf{p}$ in the matrix form, where $\mathbf{q} = \{q_j\}$, $\mathbf{P} = \{p(j|i)\}$, $\mathbf{p} = \{p_i\}$. From (2) and assuming an invertible matrix $\mathbf{P}$, one can easily estimate true class probabilities as

$$\mathbf{p} = \mathbf{P}^{-1} \cdot \mathbf{q}. \tag{3}$$

Equation (3) is correct under assumption that values of q_j and $p(j|i)$ are known with certainty. This can occur only if available training and new data sets are very large. However, for majority of real-life applications, the size of available data sets is limited and q_j and $p(j|i)$ can be considered as random variables whose properties should be estimated first.

A Bootstrap Method for Estimating p_i from S_L and S_U

Statistical inference using S_L and S_U can lead to the proper estimation of multinomial distributions q_j and $p(j|i)$. However, it can be difficult to obtain the distribution of p_i expressed with (3) in a closed form. In this study we are primarily interested in estimating the expected values of class probabilities p_i on S_U. Although distributions of q_j and $p(j|i)$ alone might be estimated in a straightforward manner, even the estimation of the expected value of p_i could not be done directly. Although $\mathbf{P}$ and $\mathbf{q}$ are independent, such an estimation is difficult since $E[\mathbf{P}^{-1}] \neq E[\mathbf{P}]^{-1}$. Therefore, we use the idea of bootstrap [8], which is a powerful simulation methodology for statistical inference suitable for estimating the distribution of p_i. We first describe the basic idea of bootstrap.

Given an original sample X with n examples *the bootstrap sample* X^* is obtained by randomly sampling n examples from X with replacement. Bootstrap algorithm generates a large number B of bootstrap samples X^{*1}, X^{*2} ..., X^{*B} and calculates desired statistics $s^{*b} = s(X^{*b})$ from each of them. Statistics $s(\cdot)$, for example, can represent the sample mean, but it can be an almost arbitrarily complex function such as the one expressed by (3). Properties of the statistics $s(\cdot)$ such as mean, variance or confidence intervals can be estimated from B obtained values s^{*b} that are called *the bootstrap replicates of* s.

In our problem two independent samples S_L and S_U are available for separately estimating q_j and $p(j|i)$. Given a labeled set S_L, one should properly use this set both for training a classifier and for estimating conditional probabilities $p(j|i)$. If n_L is large, the usual practice is to reserve a test set S_{test} of size n_{test} for estimating $p(j|i)$, and to train a classifier on the remaining data $S_{train} = S_L - S_{test}$. If n_L is relatively small, cross-validation [e.g., 8] is usually employed where, effectively, $n_L = n_{train} = n_{test}$ and all n_L examples are used both for training and for estimating $p(j|i)$, at the cost of a larger computational effort needed to learn a number of cross-validation classifiers. On the other side, all n_U examples from S_U can be directly used to estimate q_j.

In Table 1 we present a bootstrap algorithm for estimating class probabilities p_i. With $n_{test}^*(j, i)$ we denoted the number of examples in a bootstrap sample S_{test}^* that are predicted to be of class j when their true class is i. Similarly, with $n_U^*(j)$ we denoted the number of examples in a bootstrap sample S_U^* predicted to be of class j. The idea of the algorithm is clearly to generate two bootstrap samples, calculate the corresponding bootstrap replicates of q_j and $p(j|i)$ and then use equation (3) to determine bootstrap replicate of p_i. Finally, the estimate Ep_i of class probability p_i on S_U can be calculated as $Ep_i = (1/B)\Sigma_b p_i^{*b}$. According to [8], 100 – 200 bootstrap iterations are needed if we are interested only in Ep_i, and 500 – 1000 bootstrap iterations are needed if we are interested in the two-tailed confidence intervals of p_i.

Table 1. A bootstrap algorithm for estimating class probabilities p_j in unlabeled data

Given B, S_{test}, S_U and a classifier
$b = 0$
repeat
  Generate a bootstrap sample from n_{test} examples of S_{test} and calculate
  $p^*(j|i) = n_{test}^*(j,i) \Big/ \sum_j n_{test}^*(j,i)$ for $i, j = 1, ..., c$.
  Generate a bootstrap sample from n_U examples of S_U and calculate
  $q_j^* = n_U^*(j) \Big/ n_U$ for $j = 1, ..., c$.
  Use (3) to calculate bootstrap replicate p_i^{*b} for $i = 1, ..., c$.
  if all p_i^{*b} are within interval [0, 1]
    $b = b + 1$
  end
until $b = B$

It is important to observe that for small S_L and S_U some bootstrap samples can result in infeasible replicates p_i^*, but the proposed algorithm discards all such infeasible

replicates. For explanation, let us consider an example of two-class classification problem with an iteration of the bootstrap algorithm resulting in replicates $p^*(1|1) = p^*(0|0) = 0.8$ and $q_1^* = 0.9$. Clearly, assuming the conditional probabilities are true, even for the extreme case with all examples from S_U being from class 1, q_1 could not be larger then 0.8. As a consequence, applying (3) on a given example would result in infeasible replicates $p_1^* = 1.17$ and $p_0^* = -0.17$. Therefore, in the algorithm from Table 1 all such replicates are discarded.

Modifications of the Bootstrap Algorithm from Table 1

We propose two modifications of the algorithm from Table 1 in order to improve its speed and to obtain better estimates of class probabilities. Obtaining B bootstrap samples can become computationally expensive if data sets S_L and S_U are large. For sufficiently large data sets estimates of q_j and $p(j|i)$ become very close to their true values and the algorithm from Table 1 might not be necessary to estimate p_i. However, in practice it is often not clear how large data set is large enough and, therefore, we use a simple procedure to provide bootstrap replicates q_j^* and $p^*(j|i)$ computationally fast when S_L and S_U are large.

Let us assume a sample X contains discrete random variables $x_i \in \{1, ..., c\}$, $i = 1, ..., n$, such that f_j represents the fraction of examples with value j. Since vector $\boldsymbol{f}$ with elements $\{f_j\}$ is a sufficient statistics of X, replicate $\boldsymbol{f}^*$ of a bootstrap sample with elements $\{f_j^*\}$ has distribution $\boldsymbol{f}^* \sim Mult(n,\boldsymbol{f})/n$, where *Mult* denotes multinomial distribution with $E[f_j^*]=f_j$, $Var[f_j^*]=f_j(1-f_j)/n$. If n is large f_j^* can be approximated by a normal distribution $N(f_j, f_j(1-f_j)/n)$. In Table 2 we describe a procedure for random generation of $\boldsymbol{f}^*$ without the need for bootstrap sampling, where by $norm\_rnd(\mu, \sigma^2)$ we denoted a random generator of a normal distribution with mean μ and variance σ^2.

Table 2. A fast procedure for estimating frequencies from large bootstrap samples

Given n, $\boldsymbol{f}$, c. (c is the number of classes)
$f_1^* = norm\_rnd(f_1,\ f_1(1-f_1)/n)$
if c = 2
  $f_2^* = 1 - f_1^*$
else
  for i = 2: c − 1
    $f = f_i \Big/ \sum_{j=i}^{c} f_j$, $n^* = n(1 - \sum_{j=1}^{i-1} f_j^*)$
    $f_i^* = norm\_rnd(f,\ f(1-f)/n^*) \cdot n^*/n$
  end
  $f_c^* = 1 - \sum_{i=1}^{c-1} f_i^*$
end

For large S_{test} and S_U the algorithm from Table 1 can be modified so that replicates q_j^* and $p^*(j|i)$ are calculated directly by using the procedure from Table 2 instead of taking actual bootstrap samples. To perform this one should only calculate $\hat{q}_j$ from

S_U or $\hat{p}(j \mid i)$ from S_{test} and use these values as the corresponding sufficient statistics $\boldsymbol{f}$.

The second modification to the algorithm from Table 1 is using Laplace corrections [4] to improve bootstrap replicates q_j^* and $p^*(j \mid i)$ when S_L and S_U are small. Let us assume that probability that example of class i occurs in a sample is small. If the sample size n is also small, there is a considerable probability that the fraction f_i of examples from the rare class in the sample will be zero. In such a case, all the bootstrap samples will also have zero examples of class i, resulting in $f_i^* = 0$. As already shown [9], for certain cost matrices this can result in very poor predictions of classifier's loss. The idea of Laplace correction is to bias the fractions f_i^* towards uniform distribution. To achieve this, a simple adjustment of frequency f_i from the original sample is performed as

$$f_i = \frac{n_i + \lambda}{n + c\lambda} \tag{4}$$

where n_i is the number of examples from class i within a sample of size n, and λ is the Laplace coefficient. Laplace correction with $\lambda = 1$ is a very suitable choice that can be validated in the following way. If $n_i = 0$ it can be assumed that the true frequency f_i of class i is at one standard deviation from zero, i.e., $f_i = sqrt\{f_i(1-f_i)/n\}$. From there it follows that $f_i = 1/(n+1)$, which after replacing in (4) results in $\lambda = n/(n-c+1) \approx 1$. Finally, it should be noted that Laplace correction can be easily incorporated in both procedures from Table 1 and Table 2.

3 Improving Classification Based on Class Probability Estimates

Once class probabilities on S_{new} are estimated it should be possible to improve the initial classifier. We analyze two distinct cases depending on the type of classifier and on the available computational resources.

Improving a Classifier That Estimates Posterior Class Probabilities

If a classifier was trained to estimate posterior class probabilities $p(i \mid \mathbf{x})$ when presented with a new example $\mathbf{x}$, then it can be directly adjusted without the need for retraining according to estimated class probabilities p_i on S_U. For example, a neural network with a hidden layer and c outputs (representing each of the c classes) trained by minimizing the mean squared error is known to approximate posterior class probabilities [1]. Denoting class frequencies in training set S_{train} as $f_{i,train}$, and predictions of a classifier as $p(i \mid \mathbf{x})$, adjusted predictions $p_{adjust}(i \mid \mathbf{x})$ can be calculated as [1]

$$p_{adjust}(i \mid \mathbf{x}) = \frac{p(i \mid \mathbf{x}) Ep_i / f_{i,train}}{\sum_j p(j \mid \mathbf{x}) Ep_j / f_{j,train}} \tag{5}$$

If computational resources permit one could try to produce a number of predictions for any given input $\mathbf{x}$ using different bootstrap replicates of p_i, $i = 1, \ldots, c$ and average

the obtained values to the final prediction. However, this approach will not be considered further in this paper. If an arbitrary cost matrix C is assumed, one can further modify the existing classifier $p_{adjust}(i|\mathbf{x})$ to provide classifications that minimize the conditional classification risk [6].

Retraining a Classifier According to Estimated Class Probabilities

Retraining a classifier on $S_{retrain}$ resampled from S_{train} so that $f_{i,retrain} = Ep_i$ should lead to better accuracy on S_U regardless of the type of classification algorithm. Moreover, if a classifier represents a nonlinear discriminant function that directly provides classification such a retraining might represent the only viable choice to improve the classification accuracy. Therefore, in Table 3 we describe a simple iterative procedure for retraining of classifier that starts by training a classifier based on the original class probabilities from S_{train}. As seen from the Table 3, in the following iterations $S_{retrain}$ is resampled according to bootstrap estimate Ep_i.

Table 3. A procedure for retraining classifier using estimates of class probabilities Ep_i

Given $S_{train} = \{(\mathbf{x}_k, y_k)\}$, $k = 1, \ldots, n_{train}$, $y_k \in \{1, \ldots c\}$ and S_U
Assign $f_{i,retrain} = f_{i,train}$, ($f_{i,train}$ is the frequency of class i in S_{train})
repeat
For each k, $d_k = f_{i,retrain} / f_{i,train}$, where i is label of example $(\mathbf{x}_k, y_k)$
Normalize d_k such that $\Sigma_k\, d_k = 1$
Resample n_{train} examples $S_{retrain}$ from S_{train} according to d_k
Train a classifier on $S_{retrain}$
Produce bootstrap estimates p_i^{*b} on S_U
For each $i = 1, \ldots, c$, assign $f_{i,retrain} = (1/B)\Sigma_b\, p_i^{*b}$
until stopping criterion

Termination of the retraining procedure depends on the available computational resources. In the simplest, just one retraining might be needed to produce a satisfactory classifier adjusted for prediction on S_U. Also, if $f_{i,retrain}$ is very similar to $f_{i,train}$, it might be decided that retraining is not necessary. For example, if $f_{i,train}$ is within certain confidence interval of bootstrap estimate of p_i it can be claimed that, statistically, class probabilities in S_L and S_U are identical. Finally, if possible, the procedure should be repeated until convergence of estimated class probabilities is observed between consecutive iterations.

Few of the many modifications of the proposed procedure that can depend on an application include:

- If training set S_{train} is large the size of resampled data can be made smaller than n_{train} to speed-up the retraining without much loss of accuracy [12];
- For neural network classifiers, retaining on S_{train} and adjusting the weighting cost function according to estimated class probabilities could lead to a better accuracy then when resampling S_{train} [1];
- Using the similar reasoning as in section 2, Laplace correction can be used on $f_{i,retrain}$ to adjust it towards uniform distribution;

- If computational resources allow, instead of training a single classifier, bagging [3] can be used to train an ensemble of classifiers to improve both the classification accuracy and the estimate of class probabilities on S_U;
- Some of known methods [2,5] can be coupled with the procedure from Table 3 in a straightforward manner to produce classifiers that are optimized to an arbitrary cost matrix.

4 Experimental Results

We performed two groups of experiments to validate the proposed procedures for improving classifiers trained on data with biased class distribution. In the first group, we validated bootstrap methodology proposed in Section 2, while in the second group we applied the proposed methodology to the benchmark Waveform data set [2].

4.1. Validation of the Proposed Bootstrap Algorithm

The proposed bootstrap algorithm was examined across a wide range of possible scenarios including different choices of data sizes n_{test}, n_U, number of classes c, prior class probabilities $p_{i,train}$ on S_{train} and p_i on S_U, and conditional probabilities $p(j|\, i)$ describing classifier's performance. In our experiments we have first chosen several sets of parameters $\{c,\ \lambda,\ n_{test},\ n_U\}$ as shown in Table 4. Then, for each such set of parameters we randomly generated 1000 probabilities $p_{i,train}$, p_i, and $p(j|\, i)$ to obtain 1000 7-tuples $\{c,\ \lambda,\ n_{test},\ n_U,\ p_{i,train},\ p_i,\ p(j|\, i)\}$. To examine a large range of possible choices we used the following random generators (by *rand* we denote a uniform random number from [0,1]): (a) $p_{i,train} = r_i/\Sigma r_i$, where $r_i = rand + 0.05$; (b) $p_i = r_i/\Sigma r_i$, where $r_i = rand + 0.05$; and (c) $p(j|\, i) = r_{ji}/\Sigma_j r_{ji}$, where $r_{ji} = rand + \delta_{ji}$. The reason for adding 0.05 in (a) and (b) was to avoid examination of extremely rare classes in training or in unlabeled data, while adding Kronecker's delta promoted higher probabilities at $i = j$ which is behavior expected of any classifier.

In our methodology for bootstrap validation, starting from a given 7-tuple $\{c,\ \lambda,\ n_{test},\ n_U,\ p_{i,train},\ p_i,\ p(j|\, i)\}$, we use $\{n_{test},\ p_{i,train},\ p(j|\, i)\}$ to randomly generate a realization $\hat{p}(j\,|\,i)$, and $\{n_U,\ p_i,\ p(j|\, i)\}$ to randomly generate a realization $\hat{q}_j$ by the procedure described in Table 2. Therefore, for the each given 7-tuple we generate a pair $\{\ \hat{p}(j\,|\,i)\ ,\ \hat{q}_j\ \}$. Then, from $\{\ \hat{p}(j\,|\,i)\ ,\ \hat{q}_j\ \}$ we generate 500 bootstrap replicates of $\hat{p}(j\,|\,i)$ and $\hat{q}_j$, and apply (3) to calculate 500 bootstrap replicates p_i^*. Using 500 replicates p_i^* we calculate 90% confidence intervals for bootstrap estimates of p_i, and measure if all actual values p_i, $i = 1, \ldots, c$, belong to the estimated confidence intervals. If the proposed bootstrap algorithm is well designed, in about 90% of experiments the true values of p_i will belong to the estimated 90% confidence intervals.

In Table 4, for 16 different sets of parameters $\{c,\ \lambda,\ n_{test},\ n_U\}$, we report *B90* values showing the fractions of the 1000 90% confidence intervals that contained the true p_i values. As could be seen, *B90* values were between 0.83-0.92 in different

experiments, they were slightly larger for $\lambda = 1$, and did not depend much on the choice of data size and the number of classes. Slightly lower B90 means that the estimated confidence intervals for the proposed bootstrap method are slightly shorter then their true value, but the difference is small enough to conclude that the evaluated bootstrap method gives satisfactory estimates. We also report the average number of bootstrap iterations *total_b* needed to obtain 500 feasible bootstrap replicates of p_i. As seen, for c = 5 and $n_{test} = 250$, the number of infeasible solutions was very large. In other cases *total_b* was acceptably small.

Table 4. For each of 16 tuples $\{c, \lambda, n_{test}, n_U\}$, 1000 7-tuples tuples $\{c, \lambda, n_{test}, n_U, p_{i,train}, p_i, p(j|i)\}$ were generated and for each of them 500 bootstrap replicates of p_i were calculated. The fraction of the 90% confidence interval that contained the true p_i values is denoted as *B90*.

Classes	λ	n_{test}	n_U	*total_b*	*B90*
2	0	100	100	657	0.87
		100	1000	603	0.83
		1000	100	526	0.84
		1000	1000	510	0.83
	1	100	100	638	0.89
		100	1000	579	0.86
		1000	100	527	0.86
		1000	1000	507	0.85
5	0	250	250	2240	0.87
		250	5000	2110	0.85
		5000	250	976	0.84
		5000	5000	621	0.85
	1	250	250	2210	0.87
		250	5000	2070	0.92
		5000	250	936	0.87
		5000	5000	612	0.84

4.2. Experiments on Waveform Data Set

As proposed by Breiman [2] we generated waveform data sets with arbitrary number of examples with 21 continuous attributes from each of the 3 classes, where each class represents a certain combination of 2 "base" waves chosen from the 3 available “base” waves. The concept to be learned is highly nonlinear and noisy and its Bayes error is known to be 13.2% [2] if all 3 classes have the same probability in training and in test data. The fact that we were able to generate an arbitrary number of examples allowed us to perform a range of experiments with different data sizes and different class probabilities.

In the performed experiments we first defined a set of two parameters $\{n_{test}, \mathbf{p}\}$, where elements of $\mathbf{p}$, p_i, represented the class probabilities on the unlabeled data. We assumed that the class probabilities on labeled data were the same, $p_{i,train} = 1/3$. For simplicity, we also assumed a labeled data set is composed of separate training set S_{train} and test set S_{test} with the same numbers of examples n_{test}. We could as well

decided to apply a more data-efficient cross-validation approach by using S_{train} both for training and test, but the choice of the separate test set allowed us to perform a larger set of experiments.

Based on the given values $\{n_{test}, \mathbf{p}\}$ we first generated balanced sets S_{train} and S_{test} with the same number of examples n_{test}. Then, we generated $n_U = n_{test}$ (again, for convenience) examples according to class probabilities $\mathbf{p}$. Note that the exact fractions of classes within S_U can be considered as random numbers from $Mult(n_u, \mathbf{p})/n_u$ distribution. We also generated another large data set S_{large} with 30,000 examples from each of the 3 classes to be able to precisely measure the accuracy of each constructed classifier.

All combinations $\{n_{test}, \mathbf{p}\}$ used in our experiments are shown in Table 5. For each combination $\{n_{test}, \mathbf{p}\}$ we performed 30 experiments using neural networks with 21 inputs, 5 hidden nodes and 3 sigmoid outputs, trained by resilient backpropagation algorithm [10] on S_{train}. Then, we tested neural network performance on S_{test} to estimate matrix $\mathbf{P}$ with elements $p(j|i)$, and on S_U to estimate vector $\mathbf{q}$ with elements q_j. Then, we applied the proposed bootstrap method with 200 iterations to estimate vector $\mathbf{p}$ with elements p_i.

In Table 5 we report the average bootstrap estimate Ep_i and +/- one standard deviation over 30 experiments and, in the parentheses, the average length of 90% confidence intervals of p_i over 30 experiments. We also report on the average accuracy +/- one standard deviation of the original neural classifier, of the adjusted neural network using (5), and of the neural network retrained according to estimated class probabilities of unlabeled data. We were able to use (5) since described neural networks were trained to approximate posterior class probabilities $p(i|\mathbf{x})$. Note that all accuracies were measured on S_{large} by accounting for the true class probabilities of the unlabeled data $\mathbf{p}$.

Table 5. Accuracy and class probability estimates over different combinations $\{n_{test}, \mathbf{p}\}$

p	n_{test}	Bootstrap estimates of class frequencies on unlabeled data [%]			Accuracy [%]		
		Class1	Class2	Class3	Original	Adjusted	Retrain
$\frac{1}{3},\frac{1}{3},\frac{1}{3}$	500	34±3 (12)	33±3 (11)	33±3 (11)	**83.2±1.2**	**83.0±1.2**	81.1±0.9
	2000	33±2 (6)	34±2 (5)	33±2 (5)	**85.1±0.7**	**85.1±0.7**	**84.8±0.5**
$\frac{1}{2},\frac{1}{3},\frac{1}{6}$	500	49±4 (13)	33±3 (11)	18±3 (10)	82.7±1.6	**84.1±1.4**	82.4±1.5
	2000	49±2 (6)	34±2 (6)	17±2 (5)	84.1±0.8	**85.6±0.8**	**85.5±1.1**
$\frac{2}{3},\frac{1}{3},\frac{1}{6}$	500	66±5 (14)	17±3 (11)	17±4 (11)	81.1±2.3	**85.0±2.0**	83.9±1.8
	2000	67±2 (7)	17±3 (5)	17±4 (5)	83.2±1.3	86.7±1.3	**87.5±1.0**
$\frac{3}{4},\frac{1}{6},\frac{1}{12}$	500	75±4 (15)	17±4 (12)	9±2 (10)	80.0±2.5	**87.4±2.0**	86.0±1.6
	1000	75±3 (10)	17±2 (8)	8±2 (7)	81.5±1.8	**88.1±1.2**	**88.2±1.1**
	2000	75±2 (7)	17±1 (5)	8±1 (5)	81.3±1.8	88.2±1.4	**89.1±1.0**
	5000	75±1 (4)	17±1 (3)	9±1 (3)	82.4±1.7	88.9±1.2	**90.0±0.9**

As expected, for $\mathbf{p} = [1/3,1/3,1/3]$ an original neural network had the highest accuracy for n_{test}=500, while for n_{test}=2000 all 3 strategies gave similar accuracy. For small data sets the estimates of $\mathbf{p}$ can be slightly off which can cause slight decrease in the accuracy of adjusted classifier. If another classifier is trained on resampled data

using inaccurate estimate of **p** this difference can be even higher. For other 3 examined vectors **p** ([1/2,1/3,1/6], [2/3.1/3,1/6], [3/4,1/6,1/12]) adjusted classifiers and retrained classifiers were clear winners over original classifiers. Also, for large data sets retrained classifiers were superior to adjusted ones, while adjusted classifiers were better with smaller samples. Observe also that the classification accuracy was increasing with the size of the data sets for all 3 scenarios (Original, Adjusted, and Retrain). Looking at the estimated class probabilities of **p** expressed as $100p_i$ %, it can be seen that these estimates were consistent with the true class probability on unlabeled data, while the confidence intervals of these estimates decreased as the data size increased.

5 Conclusions

If class distribution on labeled data is different from that of unlabeled data, a classifier trained on labeled data can cause wrong inference and produce sub-optimal classification on unlabeled data. In this paper we proposed a bootstrap algorithm to estimate class probabilities of unlabeled data and used these estimates to improve classification on unlabeled data. It was shown experimentally that this approach can be successfully applied to improve classifiers trained on data with biased class distribution. It is worth noting that the proposed bootstrap methodology alone could be very useful in estimating the confidence intervals for class probabilities in important real-life problems such as determining the commonness of protein disorder in nature.

Although bootstraping is known as a computer intensive methodology, it is computationally fairly cheap in the framework of classification. The proposed algorithm uses an existing classifier and the estimate of its accuracy that should both be products of a standard process of classifier construction. The proposed bootstrap algorithm requires only an additional pass through unlabeled data to estimate the probability of predicting each of the classes. Based on these estimates, the bootstrap estimation of the true class probabilities on unlabeled data could be done in seconds regardless of the sizes of labeled and unlabeled data. Even when retraining a classifier to improve the prediction accuracy, the overall computational effort is just twice more costly as compared to training a single classifier.

However, it should be remembered that the proposed methodology is applicable only to the sampling bias in class distribution. Therefore, some theoretical or empirical evidence about sampling bias in labeled and unlabeled data should validate the methodology application. The goal of our work in progress is to derive statistical tests that could determine: (*i*) if class distribution is biased, and (*ii*) if there are other types of sampling bias in data. If such tests were derived, the proposed methodology could become a standard off-the-shelf procedure for machine learning and knowledge discovery.

References

1. Bishop, C.: Neural Networks for Pattern Recognition. Clarendon Press, Oxford (1995)
2. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. The Wadsworth International Group (1984)
3. Breiman, L.: Bagging predictors. Machine Learning. **24** (1996) 123-140
4. Cestnik, B.: Estimating Probabilities: A Crucial Task in Machine Learning. Proceedings of the 9th ECAI. Stockholm, Sweden (1990) 147-149
5. Domingos, P.: MetaCost: A General Method for Making Classifiers Cost-Sensitive. Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining. San Diego, ACM Press (1999) 155-164
6. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. John Wiley & Sons, New York, US (1973)
7. Dunker A.K., Lawson J.D., Brown C.J., Romero P., Oh J.,Oldfield C.J., Campen A.M., Ratlif, Hipps K.W., Ausio J., Nissen M.S., Reeves R., Kang C.H., Kissinger C.R., Bailey R.W., Griswold M.D., Chiu W., Garner E.C. and Obradovic Z.: Intrinsically Disordered Proteins. Journal of Molecular Graphics and Modeling, **19** (2001) 28-61
8. Efron, B., and Tibshirani, R. J.: An Introduction to the Bootstrap. New York: Chapman & Hall (1993)
9. Margineantu, D.D., Dietterich, T.G.: Bootstrap Methods for the Cost-Sensitive Evaluation of Classifiers. Proceedings of the 17th International Conference on Machine Learning, (2000) 582-590
10. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: the RPROP algorithm. Proceedings of the IEEE International Conference on Neural Networks. (1993) 586-591
11. Romero, P., Obradovic, Z., Li, X., Garner, E., Brown, C.J., Dunker, A.K.: Sequence Complexity and Disordered Protein. Proteins: Structure, Function and Genetics. **42** (2001) 38-48
12. Vucetic, S., Obradovic, Z.: Performance Controlled Data Reduction for Knowledge Discovery in Distributed Databases. Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining. Computer Science Editorial 3, Springer-Verlag, Kyoto, Japan (2000) 29-39
13. Vucetic, S., Radivojac, P., Dunker, K., Brown, C., Obradovic, Z.: Methods for Improving Protein Disorder Prediction. Proceedings of the IEEE/INNS International Conference on Neural Networks. Washington D.C. (2001, in press)