Neural Comput & Applic (2007) 16:295–306
DOI 10.1007/s00521-007-0089-7

ORIGINAL ARTICLE

A data reduction approach for resolving the imbalanced data issue in functional genomics

Kihoon Yoon · Stephen Kwek

Received: 1 December 2006 / Accepted: 21 December 2006 / Published online: 15 March 2007

Abstract Learning from imbalanced data occurs frequently in many machine learning applications. One positive example to thousands of negative instances is common in scientific applications. Unfortunately, traditional machine learning techniques often treat rare instances as noise. One popular approach for this difficulty is to resample the training data. However, this results in high false positive predictions. Hence, we propose preprocessing training data by partitioning them into clusters. This greatly reduces the imbalance between minority and majority instances in each cluster. For moderate imbalance ratio, our technique gives better prediction accuracy than other resampling method. For extreme imbalance ratio, this technique serves as a good filter that reduces the amount of imbalance so that traditional classification techniques can be deployed. More importantly, we have successfully applied our techniques to splice site prediction and protein subcellular localization problem, with significant improvements over previous predictors.

Keywords Data reduction · Clustering · Imbalanced data · Neural network

K. Yoon (✉) · S. Kwek
Department of Computer Science,
University of Texas at San Antonio,
San Antonio, TX 78249, USA
e-mail: kyoon@cs.utsa.edu

S. Kwek
e-mail: kwek@cs.utsa.edu

1 Introduction

Recent technological advances enable biologists to collect huge amount of genomic data by using automated DNA sequencers, microarrays that generate gene expression information of an entire organism, and other high throughput techniques. These data contain valuable information that may lead to treatments of complex disease and improve our quality of life. Although, in principle, machine learning techniques can serve as indispensable tools for analyzing genomic data, some surveys indicated that the results are far from ideal [1]. In many genomic applications, we are faced with the challenging issue of extremely high imbalanced data where we may see one positive instance (e.g. splice site) only after having seen thousands of negative instances. Henceforth, we refer to the minority class as the target class throughout the rest of the paper, with the assumption that there is only one minority class of interest. In a typical imbalanced data problem, it is more important to correctly classify the minority class. For example, in the area of computer security, most traces in computer system logs are normal non-malicious usage, but it is more important to be able to detect the rare occasional intrusion attempts. Hence training data for building an automatic intrusion detection system is highly imbalanced and false-negative mistakes tend to be more costly.

One of the popular treatments of such imbalance is to resample the training data to obtain a more balanced number of majority and minority instances. Such resampling method makes the problem more tractable and yields good accuracy on the test instances. Unfortunately, as the classifiers are constructed using the resampled data that has a distribution quite different from the actual real-world imbalance distribution, their actual prediction accuracy can be far worse than the cross-validated accuracy based on the resampled data. It is worth to note that the imbalanced dataset problem may actually arise from two different problems [2, 3]. The first is the problem of interclass imbalance where the distribution of class labels varies widely. The second problem is within-class (also called intraclass) imbalance. Within-class imbalance may occur when the members of a class are not distributed in a unimodal distribution. When resampling techniques are applied to fix the imbalance dataset problem, within-class imbalance is often ignored. In fact, resampling techniques may worsen within-class imbalance [2].

To remedy this problem, we proposed an ensemble sampling technique based on a new supervised clustering algorithm. Our technique partitions the training instances into clusters with high majority class purity. We observed that many of the clusters consist of only majority instances. This allows us to identify regions in the instance space that probably do not contain any minority training instances. Thus, we only need to construct predictive model for the other regions with lower imbalance ratio (and hence lower in-class imbalance).

More importantly, unlike traditional undersampling method, we used all the majority instances so there is no information loss. We studied the performance of our proposed technique using data sets from the UCI depository [4]. Many learning techniques (e.g. Error Correcting Output Codes, one-vs-all and all pairs voting) for multi-class problems involve the conversion of multi-class problem to a collection of binary class problems. The binary class version also tends to suffer from the data imbalance problem. Since this work will help to improve such multi-class prediction problems, we converted some of the multi-class data to binary-class data using one class against all the remaining classes. Our imbalance reduction algorithm performs better than the results from a standard machine learning technique with reduced data sets.

Encouraged by the result, we proceed to test our algorithm on real-world datasets that are highly imbalanced. To this end, we selected two functional genomic problems, namely splice site prediction and protein subcellular localization (PSL) problems. Briefly, in the splice site prediction problem, a gene can be viewed as a sequence of four letters (nucleotides) A, G, T and C. Each gene consists of alternating segments of intron and exon regions (see Fig. 1). During the transcription process, the introns are spliced out and discarded while the exons are concatenated to form the messenger RNA (mRNA). As the name suggests, splice site prediction problem is to determine where splicing occurs. For human genome, the imbalance ratio of splice sites to non-splice sites is extremely high, possibly one to many thousands. Hence, splice site prediction is a very important problem as it is the first step toward a reliable automated gene finding system. The mature mRNA after splicing goes through a translation process to produce protein. The protein is then transported to its designated subcellular location to perform its function or interact with other proteins. This information provides valuable clues to the function of proteins and their interactions. The PSL problem is to determine a destination or possibly multiple destinations of protein. As there are probably more than ten major locations, it is a multi-class problem. Like other multi-class problems, the number of instances for each destination is not equal. For example, mitochondrion has very few known proteins whereas cytoplasm and cell membrane proteins are abundant. Upon close inspection, we find that PSL predictors tend to severely under-classify the minority class for the sake of achieving higher overall accuracy. Our proposed approach minimizes this problem significantly.

2 Related work

Most learning algorithms for constructing classifiers tend to focus on obtaining high accuracy on the observed labeled training data [5]. To further aggravate this difficulty,

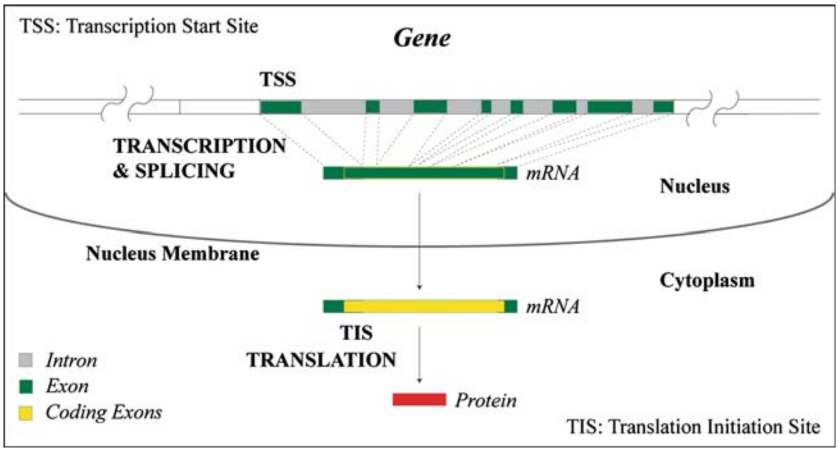

Fig. 1 Illustration of a general gene structure and genetic information flow in eukaryotic cell

almost all algorithms tend to follow the Occam's razor principle [6] (or related minimum description length MDL principle) where there is a preference toward simple hypothesis [7]. Short decision trees and neural networks with small weights are preferred [8]. The underlying assumption here is that events (instances) that occur infrequently are considered as noise. This further discriminates against the minority class so as to achieve high overall prediction accuracy. For highly imbalance data, the classifiers constructed using these algorithms predict the negative class all the time and achieve almost 100% accuracy! This is nonsensical for applications in functional genomics (and computer security) where the aim is to detect minority instances within a certain reasonable tolerance of false positive mistakes.

There are two approaches of dealing with imbalanced data sets. The first approach is the use of cost sensitive learning by modifying the classifier. The second method is to modify the prior probabilities of the majority and minority class in the training set by changing the number of instances. In particular, undersampling refers to the process of decreasing the number of instances in the majority class while oversampling is the process of increasing the number of instances in the minority class. Such resampling has several advantages over modifying the classification algorithm and cost sensitive learning. As showed in a previous work [2], a modified classification algorithm that is tailored to a specific application may not be applicable to a wide variety of problems. It is better to design a method that can handle a general class of learning problems with highly imbalanced data. Thus, resampling is a simple and attractive alternative to modifying the classification algorithm and cost sensitive learning. Various approaches [2, 3, 9–20] have been proposed to tackle the challenge posed by the imbalance ratio problem. These approaches fall into two different categories [21], namely weighting or resampling based methods. Weighting methods either assign heavier weights to the minority training instances or penalties for misclassifications of minority instances [13–15]. The other way is to preprocess training data to minimize discrepancy between class sizes. Oversampling [16] the minority class and undersampling [17, 18] the majority class are the data level approaches. Ling and Li [19] combined oversampling and undersampling methods but did not achieve significant improvement in the ''lift index'' metric. Both methods effectively change the training distribution to one that no longer resembles the original (highly imbalance) distribution, resulting in overfitting. Estabrooks et al. [20] constructed an ensemble of classifiers based on undersampling and oversampling approaches. Given a test instance, they then select for each approach, those classifers that have good fitness value and combine their predictions.

Other related works similar to resampling approaches are to focus on solving small disjuncts problem within each class [3]. Japkowicz [2] claims that the cause for standard classifiers in imbalanced data sets is due to small disjuncts (within-class imbalance). Our experiments confirm this hypothesis.

Indeed, we have identified two factors that characterize the difficult of learning with imbalanced data. The first is the complexity of the target concept's boundary. A linear boundary is easier to learn than a boundary described by a high-degree polynomial. Generally, high boundary complexity requires more instances near the boundary to give a good boundary approximation. The problem with undersampling is that the resulting sample may have too few 'majority' instances near the boundary to approximate the boundary well. This is especially so when a large portion of the majority instances are far from the boundary. In this case, using standard undersampling technique will result in a balanced sample that does not have enough boundary majority instances to tightly constrain the approximation. In fact, strong boundary complexity can even impede learning without severe class imbalance. The other factor is whether the majority instances form dense clusters. If the majority instances can be clustered into a few dense clusters then we say the majority instances have low in-class complexity. If the majority instances are widely dispersed which do not form natural clusters then they are said to have high in-class complexity. In a classification problem, classes with high in-class complexities are more difficult to label correctly than those with low in-class complexities. Maloof [22] showed that if there are big differences in the in-class complexities among classes, then the standard classifiers might not be optimized. In this scenario, the problem is actually 'the imbalance of the class sparseness' rather than the imbalanced data problem.

3 Proposed approach

The intuition behind our approach is to filter out regions in the instance space that we believe to consist of almost entirely of majority instances and hence allow us to focus on the remaining space. This allows us to reduce the imbalance ratio and hence make the learning task more tractable. Since minority instances are scarce, it is very crucial that any imbalance reduction procedure should not eliminate a minority instance from the original data. The idea is to find as many pure clusters of majority instances as possible that do not contain any minority instance or at most very few minority instances. Particularly, we look for majority instances that are far away from the target boundary (and hence reduce the amount of imbalance) so that we can concentrate on distinguishing the more difficult boundary instances. Thus, the key here is to find clusters that consist purely (or almost purely) of majority instances. Therefore, we developed a supervised clustering algorithm with class purity maximization function.

3.1 Effect of the dense majority instance clusters

A classifier that trains on the entire data set will encounter lots of negative (majority) instances close to the ideal boundary, simply because they are the majority class. This pushes the decision boundary toward the minority positive instances [5, 23]. When the ratio between majority and minority becomes larger, a classifier might treat minority instances as noisy (Fig. 2a). Figure 2b shows the decision boundary shifting after undersampling. Area between ideal and shifted decision boundary is responsible for false positive predictions. Unlike various undersampling techniques, clustering will split majority instances based on their distribution into meaningful clusters (Fig. 3). The instances in a good cluster, by definition, tend to lie in a tight region [24]. In this case, a classifier can find a decision boundary that favors more on minority class even though the number of majority instances is much higher. Another good characteristic is that the decision boundary of each classifier is dramatically different from each other. A negative instance that is wrongly classified as positive by a classifier may be corrected by the other classifiers with different decision boundary.

3.2 Overview

Our CPM algorithm selects a pair of minority and majority instances as centers as described in Algorithm 1 in Sect. 3.3. The instances are partitioned into two subsets according to their nearest centers. This process is repeated recursively for each of the two subsets until we can no longer form two clusters or no child cluster yielding higher class purity than its parent cluster. A collection of

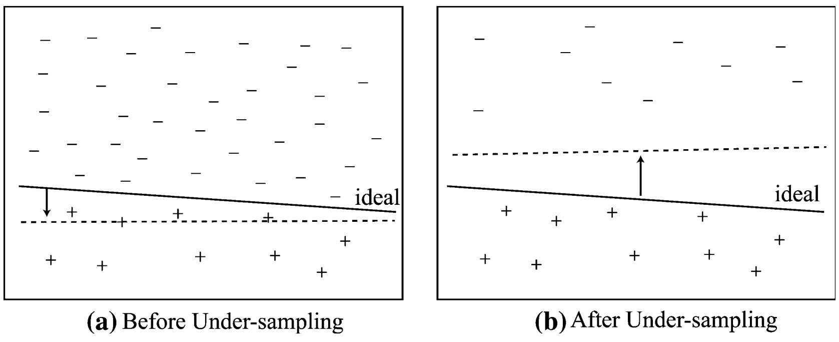

Fig. 2 Illustration of imbalanced data and undersampling: **a** imbalanced data set—decision boundary is shifted toward minority class; **b** after undersampling—decision boundary moves to majority class. The *solid line* refers an ideal decision boundary and the *dashed line* indicates an actual decision boundary estimated by a learning algorithm

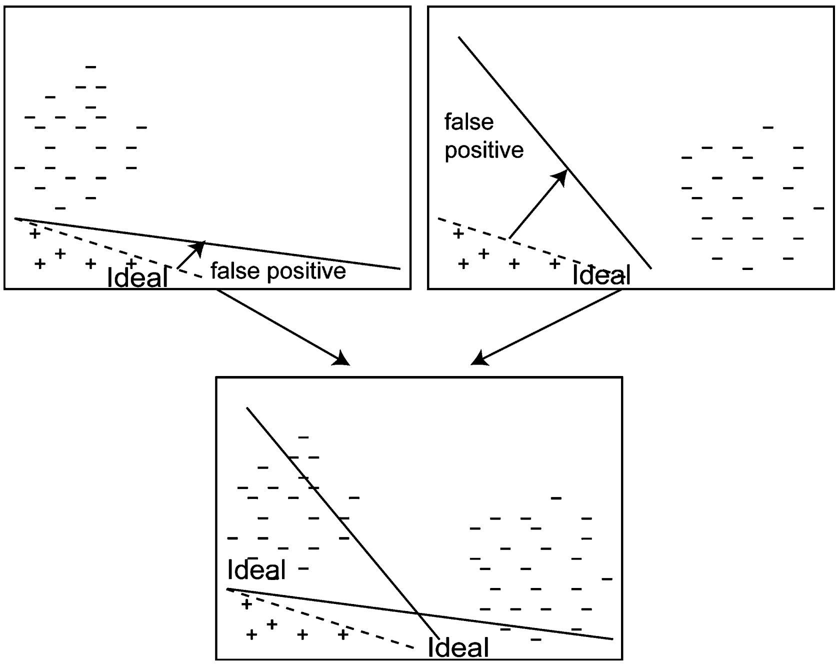

Fig. 3 Effect of small and dense subsets—give more space to minority class. Any instances placed between relaxed decision boundary and minority instances will be predicted as a minority class

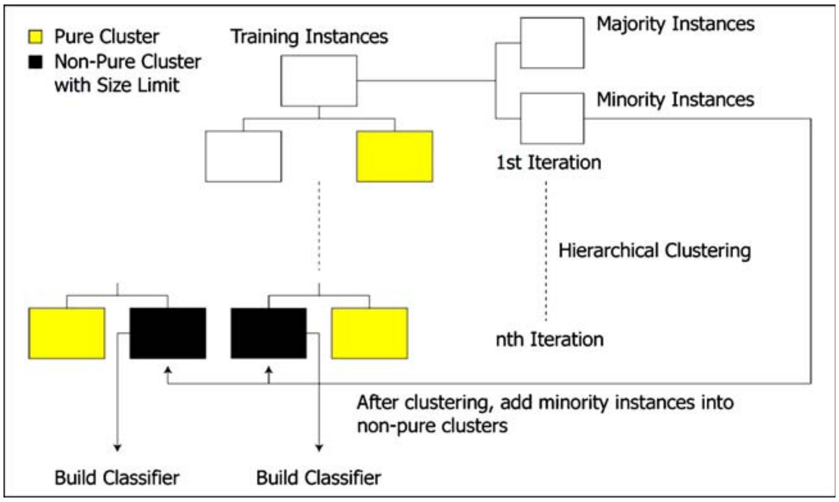

Fig. 4 Overall procedure for imbalance reduction and final decision is made by simple weighted voting among ANNs trained on non-pure clusters

samples is then constructed by adding all minority instances to each non-pure cluster, and an ANN is built for each sample.

Figure 4 illustrates the overall imbalance reduction and classification procedure. During the training step, training instances are split into two child clusters. Instances belonging to each child cluster is then evaluated for further split. As the result of the series of clustering, the leaves in the cluster tree contain either pure majority or non-pure (mixed with minority and majority instances). Since it is unavoidable that the dispersion of minority instances throughout non-pure clusters, we add all minority instances into non-pure clusters without any duplicate. Thus, each non-pure cluster consists of entire minority and reduced number of majority instances. A classifier trained on these non-pure cluster data sets tends to make minority prediction more frequently than the classifier with undersampled data. However, the decision boundary of each classifier trained on non-pure clusters is quite different from the others and this diversity can help to correct false-positive prediction as illustrated in Fig. 4. With a given unlabeled test instance, we first run through the imbalance reduction process (i.e. CPM) to estimate the best possible cluster that it might belong. If the instance belongs to a pure majority instance cluster, we simply label it as a majority instance. Only those instances belonging to a non-pure cluster is passed on to a group of ANNs which decides the class of the given instance.

3.3 Class purity maximization (CPM) clustering

The CPM algorithm is shown in Fig. 5. The algorithm attempts to find a pair of centers, one from the minority class while the second from the majority class. Using these centers, we partition all the instances into two clusters Ch_1 and Ch_2. If either of the clusters has majority class purity higher than its parent's majority class purity (*PPurity*) then we have found the centers for subclusters. Here, the purity of a set of instances is simply the ratio between the number of majority instances over the total number of instances. It then recursively partitions each of these clusters into subclusters (in Line 26 and 27). Thus, it forms hierarchical clusters. If the purity cannot be improved, then we stop the recursion (Line 2). Another stopping criterion is the number of instances in a given cluster. This is to avoid the extreme case of having singleton clusters which always have a purity of 1. The cluster size limit used for the experiment is the number of minority instances in the training data. A pair of centroid searching is done when the pair meets the selection criteria (Line 2–4). The center selection procedure is illustrated in Fig. 6. The choice of the distance measure used is the Euclidean distance.

Although the results are not shown here, exhaustive centroid pair searching gives slightly better results. Figure 7 shows the partition procedure and purity measure used for this clustering. CPM is quite different from other unsupervised clustering algorithms, in the sense that CPM uses the class labels to decide how to partition the instances. CPM does not estimate the parameters of the mixture of Gaussian distributions.

3.4 Weighted voting for final decision

The predictions from ANNs trained on non-pure clusters are weighted according to their instance distributions. Each non-pure cluster is assigned a cluster weight which is the fraction between the number of instances in the cluster (Cn_i) and total number of instances in non-pure clusters (Tn) from training data. Thus, the weight of a non-pure cluster i, W_{cluster_i} is expressed as

Fig. 5 The class purity maximization (CPM) clustering algorithm

Algorithm 1 The Class-Purity-Maximization(CPM) algorithm

```
procedure CPM($pp, P$)                                  ▷ $pp$: Parent purity, $P$: Parent
    if $|P| \le N_{min}$ or $pp = 1$ then                   ▷ stopping criteria
        return
    end if
    $purity \leftarrow 0$
    $S_{maj} \leftarrow I_{maj}(P)$            ▷ $S_{maj}$: set of majority instances from Parent
    $S_{min} \leftarrow I_{min}(P)$            ▷ $S_{min}$: set of minority instances from Parent
    $Ch1 \leftarrow \{\emptyset\}$
    $Ch2 \leftarrow \{\emptyset\}$
    $p_1 \leftarrow 0$                                       ▷ child's purity
    $p_2 \leftarrow 0$
    while $pp > purity$ do        ▷ stops when a child's purity becomes lower
        $C \leftarrow SETCENTER(S_{min}, S_{maj})$           ▷ $C = \{c_{min}, c_{maj}\}$
        $purity \leftarrow PARTITION(C, Ch1, p_1, Ch2, p_2, P)$
        if $pp > purity$ then
            $\{S_{min}\} \leftarrow \{S_{min}\} - c_{min}$   ▷ remove a tested center from the set
            $Ch1 \leftarrow \{\emptyset\}$
            $Ch2 \leftarrow \{\emptyset\}$
            $p_1 \leftarrow 0$
            $p_2 \leftarrow 0$
        end if
    end while
    if $|Ch1| = 0$ and $|Ch2| = 0$ then               ▷ a pair of center is not exist
        return
    else
        $CPM(p_1, Ch1)$
        $CPM(p_2, Ch2)$
    end if
end procedure
```

Fig. 6 The center selection procedure

Algorithm 2 The Center Selecting Procedure

```
procedure SetCenter($minInsts, majInsts$)
    $max \leftarrow 0$
    $maxIndex \leftarrow \{\emptyset\}$
    for $i \leftarrow 1, |minInsts|$ do                ▷ $d()$: Euclidean distance measure
        for $j \leftarrow 1, |majInsts|$ do
            if $max < d(minInsts_i, majInsts_j)$ then
                $max \leftarrow d(minInsts_i, majInsts_j)$
                $maxIndex \leftarrow (i, j)$
            end if
        end for
    end for
    return $(minInsts_{maxIndex1}, minInsts_{maxIndex2})$
end procedure
```

$$W_{\text{cluster}_i} = 1 + \frac{\text{Cn}_i}{\text{Tn}}. \qquad (1)$$

In addition to the cluster weight, each non-pure cluster also has class weights. The number of minority and majority instances in each non-pure cluster is counted before adding all minority instances into the cluster. The ratio between the number of instances of a class (Nm_t) and total number of instances in the cluster (Cn) is the class weight, W_{class_t} where $0 < t \le k$ and k is the number of classes.

$$W_{\text{class}_t} = 1 + \frac{\text{Nm}_t}{\text{Cn}}. \qquad (2)$$

The additive term in both Eq. (1) and (2) is used to avoid very small final prediction value. W_{class_t} considers number of minority or majority instances in a non-pure cluster t. Thus, each cluster holds two W_{class_t} $(W_{\text{class}_{\text{min}}}, W_{\text{class}_{\text{maj}}})$ values for binary-class problems.

As illustrated in Fig. 8, if an unlabeled test instance belongs to one of the non-pure clusters then we will use the

Fig. 7 The partition procedure

Algorithm 3 The Partition Procedure

```
1:  procedure PARTITION($C, Ch1, p_1, Ch2, p_2, P$)          ▷ $C = \{c_{min}, c_{maj}\}$
2:      $P_{max} \leftarrow 0$
3:      if $d(P_i, c_{min}) \leq max(d(minInsts_i, c_{min}))$ then
4:          $Ch1 \leftarrow P_i$                    ▷ $d()$: Euclidean distance measure
5:      else                                        ▷ $P_i$: $i$-th instance in P
6:          $Ch2 \leftarrow P_i$
7:      end if
8:      $p_1 \leftarrow |majCh1|/|Ch1|$                                  ▷ Purity measure
9:      $p_2 \leftarrow |majCh2|/|Ch2|$    ▷ $|majCh2|$: number of majority instances
10:     $P_{max} \leftarrow max(p_1, p_2)$
11:     return $P_{max}$
12: end procedure
```

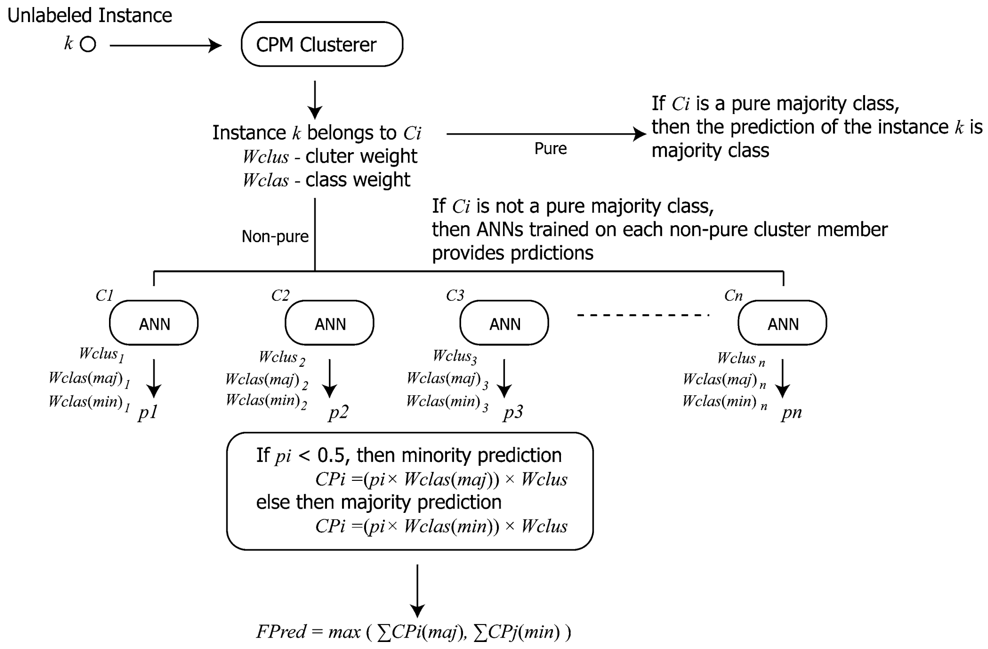
Unlabeled Instance
k
CPM Clusterer
Instance k belongs to Ci
Wclus - cluter weight
Wclas - class weight
Pure
If Ci is a pure majority class, then the prediction of the instance k is majority class
Non-pure
If Ci is not a pure majority class, then ANNs trained on each non-pure cluster member provides prdictions
C1
C2
C3
Cn
ANN
Wclus1 Wclas(maj)1 Wclas(min)1 p1
Wclus2 Wclas(maj)2 Wclas(min)2 p2
Wclus3 Wclas(maj)3 Wclas(min)3 p3
Wclusn Wclas(maj)n Wclas(min)n pn
If pi < 0.5, then minority prediction
CPi =(pi× Wclas(maj)) × Wclus
else then majority prediction
CPi =(pi× Wclas(min)) × Wclus
FPred = max (∑CPi(maj), ∑CPj(min))

Fig. 8 The illustration of prediction procedure

ANNs to predict its label. The prediction score p_i from a non-pure cluster i is weighted as follows. If p_i is a majority prediction from cluster i which is denoted as $p_{i_{\mathrm{maj}}}$, then $\mathrm{CP}_{i_{\mathrm{maj}}} = (p_{i_{\mathrm{maj}}} \times W_{\mathrm{class}_{\mathrm{maj}}})W_{\mathrm{cluster}_i}$. For a minority prediction of $p_{i_{\min}}, \mathrm{CP}_{i_{\min}} = (p_{i_{\min}} \times W_{\mathrm{class}_{\min}})W_{\mathrm{cluster}_i}$. The final decision (FPred) is made by taking max between sum of CP_i for majority and minority predictions.

$$\mathrm{FPred} = \max\left(\sum_i \mathrm{CP}_{i_{\mathrm{maj}}}, \sum_j \mathrm{CP}_{j_{\min}}\right) \qquad (3)$$

4 Experiments

In Table 1, the data sets used for our experiments are listed with a brief description. Except for the splice site and PSL data sets, all data sets were obtained from UCI Repository for Machine Learning [4]. All UCI data sets were converted to binary class problems to make their imbalance ratio as high as possible. Some of the UCI data sets, like hepatitis, balance-scale, page-blocks and primary-tumor, have very few instances. Due to the small number of minority instances in these data sets, we used fivefold cross validation. All the data sets were examined 30 times to avoid an extreme result by choosing worst instances in random resampling procedure. We applied resampling method to each training set until 20% of the data in the training set consists of the minority class. For the splice site and PSL data, we constructed two different imbalance ratio data sets from each. Thus, the imbalance ratio of our data sets ranges from 2 to 400.

4.1 Learning techniques

In this section we provide detailed descriptions of the key learning techniques used in our experiments.

Table 1 The list of data sets used for the experiments

Data sets	Imbalance (maj/min)	Minority instances	Majority instances	Total instances
Hepatitis	3.8	32	123	155
Balance-scale	11.7	44	518	562
Hypothyroid	12.0	262	3,132	3,394
Page-blocks	138.3	46	6,360	6,406
Pima-indian-diabetes	1.9	241	450	691
Primary-tumor	13.1	25	315	339
Segment	6.0	297	1,782	2,079
Sick	15.3	208	3,187	3,395
Splice site 1	100.0	300	30,000	30,300
Splice site 2	400.0	100	40,000	40,100
PSL1	20.0	45	900	945
PSL2	40.0	45	1,800	1,845

4.1.1 *Base classifier*

Our base classifier is an implementation of 'Multilayer-Perceptron', an ANN [25], in Weka machine learning suit [26]. The ANN used here is a two-layer sigmoid neural network with backpropagation rule [27]. The number of hidden nodes, H_n is set to the following default value used in Weka:

$$H_n = \frac{A_n + C_n}{2}, \tag{4}$$

where A_n is the number of attributes and C_n is the number of classes. The number of output nodes equals to C_n. All the option values are set to the default values used in Weka version 3–4. Learning rate and momentum are 0.3 and 0.2, respectively. Maximum number of iterations is set to 500. Figure 9 shows a simplified view of an ANN.

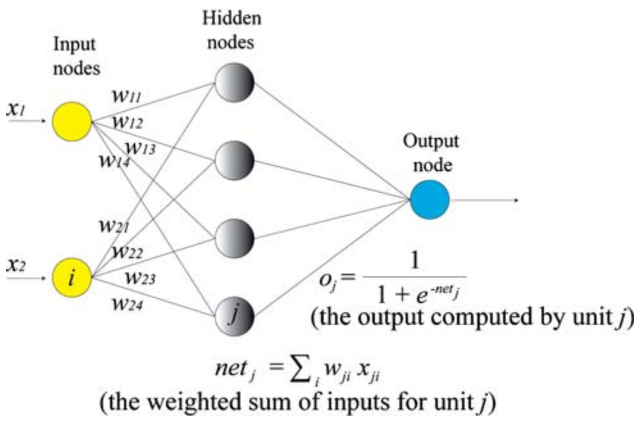
Input nodes
Hidden nodes
Output node
x_1
x_2
w_{11}
w_{12}
w_{13}
w_{14}
w_{21}
w_{22}
w_{23}
w_{24}
i
j
$o_j = \frac{1}{1+e^{-net_j}}$
(the output computed by unit j)
$net_j = \sum_i w_{ji} x_{ji}$
(the weighted sum of inputs for unit j)

Fig. 9 A simplified view of an artificial neural network

4.1.2 *Undersampling*

A straightforward approach of circumventing the data imbalance problem is to sample a smaller set of majority instances while retaining all the minority instances. This has an advantage for genomic applications where the number of majority instances is huge whereby reducing the training data also reduces the training time and makes the learning problem tractable. However, undersampling may results in lost of valuable information from the discarded instances. For our experiments, we used random undersampling for simplicity. Majority class instances were randomly removed until the imbalance ratio reaches the desired level. All undersampled data sets for this paper have imbalance ratios range from 1 to 4. One of the potential problems associated with undersampling technique is that we may discard instances that are very informative. Despite this possible limitation, random undersampling is very popular and often effective approach to deal with imbalanced data.

4.1.3 *Oversampling*

Another approach to dealing with the imbalanced data problem is to increase the number of minority class instances by oversampling them. The advantage of oversampling is that there is no information loss since there is no discarded instance. However, the minority instances are over-represented in our training set. Further, as we increase the training set, the computational cost also increases. This could result in some problems with large data sets, like those in genomic analysis application, to be intractable. There is no conclusive study with regard to whether undersampling or oversampling is better in classification accuracy. The strength of these two methods depends on both the characteristics of the data and the classification algorithms used. For our experiment, we oversample the minority instances until the training data has an imbalance ratio of less than 4.

4.2 Performance measure

The general performance measure, (estimated) test error, is not a good metric for imbalanced data. For many important bioinformatics or computer security applications, the minority instances may be less than 1% of the entire data. By simply predicting according to the majority class, we can achieve more than 99% accuracy. Clearly such predictor is not useful at all. For applications with high imbalance ratio, we frequently want to recall as many minority instances as possible. Further, we want to be precise so that when we predict an unlabeled instance to be minority class instance, there is a good chance that we are right. These two goals are often contradictory goals and we need to strike a compromise. We use *F*-measure to measure the overall performance (as the compromise between recall and precision) of the algorithms studied. The exact definitions of the recall (*R*) and, precision (*P*) were first introduced in the information retrieval community. Recall is defined as

$$R = \frac{\mathrm{CP}}{\mathrm{TP}} \tag{5}$$

where CP is the number of instances that are correctly predicted as positive and TP is the number of actual positive instances. Precision is defined as

$$P = \frac{\mathrm{CP}}{\mathrm{PP}} \tag{6}$$

where PP is total number instances predicted as positive. *F*-measure is defined as

$$F = 2 \times \left(\frac{R \times P}{R + P}\right) \tag{7}$$

which is a harmonic mean between recall and precision. *F*-measure becomes zero if either *R* or *P* is zero. It becomes 1 when both *R* and *P* are 1. Fivefold cross validation was used to estimate *R* and *P* for this paper.

5 Results and discussion

5.1 Performance comparisons among ANN with original data set and with resampled data sets

The performance of an ANN with original data and with resampling is summarized in Tables 2, 3 and 4. In general, ANN with original data has the highest precision and also shows that some data sets are not affected by the

Table 2 Performance of the artificial neural network without resampling; each performance measure is obtained by running data sets 30 times with fivefold cross validation

Data sets	ANN with original data set		
	R	*P*	*F*
Hepatitis	0.511 ± 0.011	0.536 ± 0.009	0.522 ± 0.009
Balance-scale	0.056 ± 0.016	0.217 ± 0.048	0.080 ± 0.020
Hypothyroid	0.492 ± 0.010	0.760 ± 0.020	0.589 ± 0.005
Page-blocks	0.575 ± 0.006	0.896 ± 0.009	0.699 ± 0.005
Pima-indians-diabetes	0.607 ± 0.006	0.654 ± 0.003	0.629 ± 0.003
Primary-tumor	0.783 ± 0.008	0.696 ± 0.009	0.735 ± 0.005
Segment	0.992 ± 0.001	0.986 ± 0.001	0.989 ± 0.001
Sick	0.987 ± 0.001	0.980 ± 0.001	0.984 ± 0.000
Splice site 1	0.684 ± 0.009	0.747 ± 0.012	0.713 ± 0.003
Splice site 2	0.321 ± 0.007	0.614 ± 0.016	0.422 ± 0.005
PSL1	0.387 ± 0.011	0.481 ± 0.013	0.426 ± 0.009
PSL2	0.026 ± 0.006	0.121 ± 0.020	0.040 ± 0.008

The measurements are averaged and expressed with standard error of mean

Table 3 Performance of the artificial neural network with undersampling; each performance measure is obtained by running data sets 30 times with fivefold cross validation

Data sets	ANN with undersampling		
	R	*P*	*F*
Hepatitis	0.564 ± 0.011	0.521 ± 0.007	0.540 ± 0.007
Balance-scale	0.462 ± 0.036	0.411 ± 0.018	0.427 ± 0.025
Hypothyroid	0.525 ± 0.008	0.511 ± 0.012	0.514 ± 0.005
Page-blocks	0.990 ± 0.004	0.324 ± 0.006	0.487 ± 0.007
Pima-indians-diabetes	0.721 ± 0.006	0.586 ± 0.004	0.646 ± 0.003
Primary-tumor	0.821 ± 0.008	0.539 ± 0.010	0.649 ± 0.008
Segment	0.992 ± 0.001	0.985 ± 0.001	0.988 ± 0.001
Sick	0.795 ± 0.005	0.634 ± 0.008	0.704 ± 0.004
Splice site 1	0.961 ± 0.002	0.234 ± 0.002	0.377 ± 0.002
Splice site 2	0.689 ± 0.006	0.038 ± 0.000	0.071 ± 0.001
PSL1	0.593 ± 0.011	0.287 ± 0.005	0.386 ± 0.006
PSL2	0.421 ± 0.013	0.089 ± 0.002	0.146 ± 0.004

The measurements are averaged and expressed with standard error of mean

Table 4 Performance of the artificial neural network with oversampling; each performance measure is obtained by running data sets 30 times with fivefold cross validation

Data sets	ANN with oversampling		
	R	*P*	*F*
Hepatitis	0.541 ± 0.012	0.505 ± 0.011	0.521 ± 0.010
Balance-scale	0.405 ± 0.028	0.446 ± 0.013	0.415 ± 0.019
Hypothyroid	0.637 ± 0.008	0.763 ± 0.017	0.689 ± 0.007
Page-blocks	0.964 ± 0.005	0.691 ± 0.007	0.804 ± 0.005
Pima-indians-diabetes	0.766 ± 0.006	0.567 ± 0.002	0.651 ± 0.002
Primary-tumor	0.807 ± 0.006	0.674 ± 0.010	0.733 ± 0.007
Segment	0.997 ± 0.001	0.982 ± 0.001	0.989 ± 0.001
Sick	0.730 ± 0.009	0.727 ± 0.009	0.726 ± 0.004
Splice site 1	0.859 ± 0.003	0.497 ± 0.002	0.629 ± 0.002
Splice site 2	0.521 ± 0.008	0.082 ± 0.001	0.142 ± 0.001
PSL1	0.428 ± 0.016	0.408 ± 0.012	0.417 ± 0.013
PSL2	0.136 ± 0.011	0.119 ± 0.011	0.126 ± 0.010

The measurements are averaged and expressed with standard error of mean

Table 5 *F*-measure comparisons among ANN without and with resampling; each performance measure is obtained by running data sets 30 times with fivefold cross validation

	ANN	Undersampling	Oversampling
Hepatitis	0.522 ± 0.009	**0.540 ± 0.007**	0.521 ± 0.010
Balance-scale	0.080 ± 0.020	**0.427 ± 0.025**	0.415 ± 0.019
Hypothyroid	0.589 ± 0.005	0.514 ± 0.005	**0.689 ± 0.007**
Page-blocks	0.699 ± 0.005	0.487 ± 0.007	**0.804 ± 0.005**
Pima-indians-diabetes	0.629 ± 0.003	0.646 ± 0.003	**0.651 ± 0.002**
Primary-tumor	**0.735 ± 0.005**	0.649 ± 0.008	0.733 ± 0.007
Segment	**0.989 ± 0.001**	0.988 ± 0.001	0.989 ± 0.001
Sick	**0.984 ± 0.000**	0.704 ± 0.004	0.726 ± 0.004
Splice site 1	**0.713 ± 0.003**	0.377 ± 0.002	0.629 ± 0.002
Splice site 2	**0.422 ± 0.005**	0.071 ± 0.001	0.142 ± 0.001
PSL1	**0.426 ± 0.009**	0.386 ± 0.006	0.417 ± 0.013
PSL2	0.040 ± 0.008	**0.146 ± 0.004**	0.126 ± 0.010

The measurements are averaged and expressed with standard error of mean. The number in bold face indicate the best result from three methods according to *F*-measure

imbalance in the data sets, especially segment and Sick. This suggests that these data sets are already in optimal for learning process. Modification of such data distribution could yields to suboptimal performance as shown in Tables 3 and 4. The performance of resampling techniques with ANN seems data dependent. As expected, undersampling method generally improves the recall rate over original data or oversampling at the expense of the precision rate. Interestingly, for the balance-scale data set, undersampling worked much better although the imbalance ratio is just 11.7 in contrast to the page-block data set with the imbalance ratio of 138.3. Page-block data shows significant improvement by oversampling method. Oversampling generated much better *F*-measure compared to undersampling (Table 4). In Table 5, resampling approaches, either undersampling or oversampling improves *F*-measure 6 cases out of 12 data sets. For primary-tumor and Sick data sets, ANN without resampling performs better in terms of *F*-measure. Segment data set is not affected by either resampling method. These indicate that resampling techniques are data dependent. Further, for the pima-Indians-diabetes data set, the performance of both methods is worse than primary-tumor and even though it has the lowest imbalance ratio of 1.9. This suggests boundary and in-class complexity are at work, and imbalance ratio itself does not always make the learning problem difficult.

5.2 Performance comparisons among data reduction scheme and ANNs with different resampling methods

The performance of data reduction with a simple weighted voting as shown in Table 6 is tend to be better than ANN with or without resampling. Our proposed CPM method improves the precision rate as shown in Table 6. The data reduction technique performs better on 8 data sets out of 12. This suggests that CPM managed to find meaningful clusters that can be used as filter to identify many majority instances.

Table 6 Performance of the artificial neural network with resampled data and with data reduction approach; the numbers in bold face indicate the best result from the two methods according to *F*-measure

Data sets	Best results	Data reduction (CPM)		
	F	*R*	*P*	*F*
Hepatitis	0.540 ± 0.007	0.624 ± 0.001	0.612 ± 0.002	**0.618 ± 0.001**
Balance-scale	0.427 ± 0.025	0.685 ± 0.017	0.498 ± 0.018	**0.577 ± 0.019**
Hypothyroid	**0.689 ± 0.007**	0.513 ± 0.003	0.806 ± 0.001	0.627 ± 0.005
Page-blocks	**0.804 ± 0.005**	0.752 ± 0.005	0.611 ± 0.004	0.674 ± 0.003
Pima-indians-diabetes	0.651 ± 0.002	0.704 ± 0.012	0.641 ± 0.018	**0.671 ± 0.007**
Primary-tumor	0.735 ± 0.005	0.885 ± 0.003	0.659 ± 0.006	**0.755 ± 0.002**
Segment	**0.989 ± 0.001**	0.911 ± 0.002	0.849 ± 0.001	0.879 ± 0.001
Sick	**0.984 ± 0.000**	0.894 ± 0.004	0.653 ± 0.006	0.755 ± 0.002
Splice site 1	0.713 ± 0.003	0.891 ± 0.001	0.709 ± 0.002	**0.789 ± 0.001**
Splice site 2	0.422 ± 0.005	0.865 ± 0.002	0.438 ± 0.007	**0.581 ± 0.001**
PSL1	0.426 ± 0.009	0.727 ± 0.007	0.419 ± 0.012	**0.532 ± 0.004**
PSL2	0.146 ± 0.004	0.556 ± 0.018	0.233 ± 0.001	**0.328 ± 0.003**

Each performance measure is obtained by running data sets 30 times with fivefold cross validation. The measurements are averaged and expressed with standard error of mean. The numbers in bold face indicate better results among the best result of the three methods and our approach according to *F*-measure

6 Conclusions

The results shown in Sect. 5 suggest that higher imbalance ratio by itself may not dictate the difficulty of the learning problem. Sometimes fairly balanced data such as the hepatitis data set can be more difficult to learn than highly imbalanced data such as page-blocks data. Clearly, boundary and in-class complexity are the important factors that determine the problem's difficulty. For example, if a data set is linearly separable and noise level is very low in the data, then the effect of imbalance ratio is small. In that sense, the proposed CPM approach is to reduce the imbalance ratio as well as in-class imbalance as illustrated in Fig. 3. Conceptually, each tight cluster from the majority class instances is more likely to be separable from minority class instances since majority class members in a cluster are distributed uniformly. Hence, it is possible that CPM can lower interclass imbalance as well as intraclass imbalance. More importantly, CPM can be implemented easily in a distributed system as it is easily parallelizable. This allows us to better handle very large data sets commonly found in bioinformatics applications.

Using the highly customizable characteristic of our predictor, we successfully applied our data reduction approach to an automated gene finding system, promoter prediction, and transcription factor binding site prediction as a good filter. It is possible to apply this approach to many other genomic data such as 'finding candidate genes responsible for multigenic diseases', 'predictions of sequence signals associated with mRNA stability', 'Finding point mutation on various cancer genes' and other functional genomics problems.

References

1. Ashurst J, Collins J (2003) Gene annotation: prediction and testing. Ann Rev Genomics Human Genetics 4:69–88
2. Japkowicz N (2003) Class imbalances: are we focusing on the right issue? Notes from the ICML workshop on learning from imbalanced data sets II
3. Jo T, Japkowicz N (2004) Class imbalances versus small disjuncts. ACM SIGKDD Explorat 6(1):40–49
4. Blake C, Mertz C (1998) UCI repository of machine learning databases
5. Provost F (2000) Machine learning from imbalanced data sets 101. Invited paper for the AAAI'2000 workshop on imbalanced data sets.
6. Murphy PM, Pazzani MJ (1994) Exploring the decision forest: an empirical investigation of Occam's razor in decision tree induction. J Artif Intell Res 1:257–275
7. Mitchell T (1997) Machine learning. McGraw-Hill, New York
8. Mehta M, Rissanen J, Agrawal R (1995) MDL-based decision tree pruning. In: Proceedings of the first international conference on knowledge discovery and data mining, Menlo Park, CA. AAAI Press, pp 216–221
9. Japkowicz N (2000) The class imbalance problem: significance and strategies. In: Proceedings of the 2000 international conference on artificial intelligence: special track on inductive learning, Las Vegas, NV
10. Nickerson A, Japkowicz N, Millos E (2001) Using unsupervised learning to guide resampling in imbalanced data sets. In: Proceedings of the 8th international workshop on ai and statistics, pp 261–265
11. Kotsiantis SB, Pintelas PE (2003) Mixture of expert agents for handling imbalanced data sets. Ann Math Comput Teleinform 1(1):46–55

12. Kolcz A, Alspector J (2002) Asymmetric missing-data problems: overcoming the lack of negative data in preference ranking. Informat Retr 5(1):5–40
13. Akbani R, Kwek S, Japkowicz N (2004) Applying support vector machines to imbalanced datasets. In: Proceedings of the 15th European conference on machine learning (ECML), pp 39–50
14. Domingos P (1998) How to get a free lunch: a simple cost model for machine learning applications. In: Proceedings of AAAI-98/ICML98, workshop on the methodology of applying machine learning, pp 1–7
15. Veropoulos K, Campbell C, Cristianini N (1999) Controlling the sensitivity of support vector machines. In: Proceedings of the international joint conference on AI, pp 55–60
16. Chawla N, Bowyer K, Hall L, Kegelmeyer W (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
17. Drummond C (2003) C4.5, Class imbalance, and cost sensitivity: why undersampling beats over-sampling. In: ICML-KDD'2003 workshop: learning from imbalanced data sets
18. Kubat M, Matwin S (1997) Addressing the curse of imbalanced training sets: one-sided selection. In: Proceedings of the 14th international conference on machine learning
19. Ling C, Li C (1998) Data mining for direct marketing problems and solutions. In: Proceedings of the fourth international conference on knowledge discovery and data mining, New York, NY
20. Estabrooks A, Jo T, Japkowicz N (2004) A multiple resampling method for learning from imbalanced data sets. Comput Intell 20(1)
21. Abe N (2003) Sampling approaches to learning from imbalanced datasets: active learning, cost sensitive learning and beyond. In: ICML-KDD'2003 workshop: learning from imbalanced data sets
22. Maloof M (2003) Learning when data sets are imbalanced and when costs are unequal and unknown. In: ICML-2003 workshop on learning from imbalanced data sets II
23. Provost F, Fawcett T (2001) Robust classification for imprecise environments. Mach Learn 42/3:203–231
24. Wu G, Chang E (2003) Class-boundary alignment for imbalanced dataset learning. In: ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC
25. Wasserman P (1993) Advanced methods in neural computing. Van Nostrand Reinhold
26. Witten I, Frank E (2000) Data mining: practical machine learning tools with Java implementations. Morgan Kaufmann, San Francisco
27. Chauvin Y, Rumelhart D (1995) Backpropagation: theory, architectures, and applications (edited collection). Lawrence Erlbaum, Hillsdale