Synthetic Protein Sequence Oversampling method for classification and remote homology detection in imbalanced protein data

Majid M. Beigi, Andreas Zell

University of Tübingen
Center for Bioinformatics Tübingen (ZBIT)
Sand 1, D-72076 Tübingen, Germany
{majid.beigi,andreas.zell}@uni-tuebingen.de

Abstract. Many classifiers are designed with the assumption of well-balanced datasets. But in real problems, like protein classification and remote homology detection, when using binary classifiers like support vector machine (SVM) and kernel methods, we are facing imbalanced data in which we have a low number of protein sequences as positive data (minor class) compared with negative data (major class). A widely used solution to that issue in protein classification is using a different error cost or decision threshold for positive and negative data to control the sensitivity of the classifiers. Our experiments show that when the datasets are highly imbalanced, and especially with overlapped datasets, the efficiency and stability of that method decreases. This paper shows that a combination of the above method and our suggested oversampling method for protein sequences can increase the sensitivity and also stability of the classifier. **S**ynthetic **P**rotein **S**equence **O**versampling (SPSO) method involves creating synthetic protein sequences of the minor class, considering the distribution of that class and also of the major class, and it operates in data space instead of feature space. We used G-protein-coupled receptors families as real data to classify them at subfamily and sub-subfamily levels (having low number of sequences) and could get better accuracy and Matthew's correlation coefficient than other previously published method. We also made artificial data with different distributions and overlappings of minor and major classes to measure the efficiency of our method. The method was evaluated by the area under the Receiver Operating Curve (ROC).

1 Introduction

A dataset is imbalanced if the classes are not equally represented and the number of examples in one class (major class) greatly outnumbers the other class (minor class). With imbalanced data, the classifiers tend to classify almost all instances as negative. This problem is of great importance, since it appears in a large number of real domains, such as fraud detection, text classification, medical diagnosis and protein classification [1, 2]. There have been two types of

solutions for copping with imbalanced datasets. The first type, as exemplified by different forms of re-sampling techniques, tries to increase the number of minor class examples (oversampling) or decrease the number of major class examples (undersampling) in different ways. The second type adjusts the cost of error or decision thresholds in classification for imbalanced data and tries to control the sensitivity of the classifier [3–6].

Undersampling techniques involve loss of information but decrease the time of training. With oversampling we do not loose the information but instead it increases the size of the training set and so the training time for classifiers. Furthermore, inserting inappropriate data can lead to overfitting. Some researchers [2] concluded that undersampling can better solve the problem of imbalanced datasets. On the other hand, some other researchers are in favor of oversampling techniques. Wu and Chang [7] showed that with imbalanced datasets, the SVM classifiers learn a boundary that is too close to positive examples. Then if we add positive instances (oversampling), they can push the boundary towards the negative data, and we have increased the accuracy of classifier.

To decide the question of oversampling vs. undersampling, two parameters should be taken into consideration: the *imbalance ratio* and the distribution of data in imbalanced datasets. The *imbalance ratio* ($\frac{NumberOfMinorityData}{NumberOfMajorityData}$) is an important parameter that shows the degree of imbalance. In undersampling we should be sure of the existence of enough information in the minor class and also of not loosing the valuable information in the major class. We found out that the oversampling technique can balance the class distribution and improve that situation. But the distribution of inserted positive instances is of great importance. Chawla et al. [8] developed a method for oversampling named Synthetic Minority Oversampling Technique (SMOTE). In their technique, between each positive instance and its nearest neighbors new synthetic positive instances were created and placed randomly between them. Their approach proved to be successful in different datasets.

On the other hand Veropoulos et al. [6] suggested using different error costs (DEC) for positive and negative classes. So the classifier is more sensitive to the positive instances and gets more feedback about the orientation of the class-separating hyperplane from positive instances than from negative instances.

In protein classification problems the efficiency of that approach (Veropoulos et al. [6]) has been accepted. In kernel based protein classification methods [9, 10, 1] a class-depending regularization parameter is added to the diagonal of the kernel matrix: $K'(x,x) = K(x,x) + \lambda n/N$, where n and N are the number of positive (or negative) instances and the whole dataset, respectively. But, based on our experiments, if the dataset is highly imbalanced and has overlapping data, choosing a suitable ratio of error costs for positive and negative examples is not always simple and sometimes the values near the optimum value of the error cost ratio give unsatisfying results.

We propose an oversampling technique for protein sequences in which the minority class in the data space is oversampled by creating synthetic examples. Working with protein data in data space instead of feature space allows us to

consider the probability distribution of residues of the sequence using a HMM (Hidden Markov Model) profile of the minority class and also one of the majority class and then synthesize protein sequences which can push precisely the boundary towards the negative examples. So we increase the information of the minor class. Our method of oversampling can cause the classifier to build larger decision regions for the minor class without overlapping with the major class. In this work we used real and artificial data with different degrees of overlapping and imbalance ratio to show the efficiency of our methods and we also suggest that our algorithm can be used along with DEC methods to increase the sensitivity and stability of the classifier. As SVM classifiers and kernel methods outperformed other methods in protein classification [9, 1, 10], we discuss the efficiency of our oversampling technique when used with kernel-based classifiers.

2 SPSO: Synthetic Protein Sequence Oversampling technique

Given a set of positive training sequences (minor class) S_+ and a set of negative training sequences (major class) S_- we want to create synthetic protein sequences $S_{synthetic}$ as mutated replicas of each sequence of the minor class, provided that those synthetic sequences are created by an HMM profile (Hidden Markov Model profile) of the minor class and are phylogenetically related to that class and far away from the major class. For this, at first we build a multiple alignment of the sequences of the minor class using ClustalW [11] and then we train a hidden Markov model profile with length of the created multiple alignment sequences for each class (positive data and every family belonging to the negative data).

For every sequence in the minor class we create another mutated sequence synthetically. For that, we consider an arbitrary N_m as number of start points for mutation in that sequence. We suppose the $HMMp_+$ (hidden Markov model profile of positive instances) has emitted another sequence identical to the main sequence until the first point of mutation. From that point afterward we assume that $HMMp_+$ emits new residues until the emitted residue is equal to a residue in the same position in the main sequence. From this residue, all residues are the same as residues in the original sequence until the next point of mutation (Fig. 1).

In this way, if the point of mutation belongs to a low entropy area of the HMM profile the emitted residue will be very similar to the main sequence (will have few mutations). We expect the emmitance probability of the synthesized sequence with $HMMp_+$ to be higher than with $HMMp_-$, if not (very rarely), we synthesize another one or we decrease the value of N_m. The N_m parameter can adjust the radius of the neighborhood of the original sequences and the synthesized sequences. With larger values of N_m, the algorithm creates sequences that are phylogenetically farer away from main sequences and vice versa. We used another routine to find a suitable value of N_m. At first, in the minor class, we find the protein sequence which has the highest emission probability with the HMM

Algorithm SPSO(S_+,S_-)

Input : S_+, set of sequences of minority class; S_-, set of sequences of majority class

Output: $S_{synthetic}$, set of synthetic protein sequences from the minority class

Create HMM profile of set S_+, call it $HMMp_+$;
Create array of HMM profiles consisting of all families belonging to S_-, call it $HMMp_-[]$;
Choose an arbitrary number as number of start points for mutation, call it N_m;
for $i \leftarrow 1$ **to** $|S_+|$ **do**
    $s = S_+[i]$;
    repeat
        Create an array of sorted non-repeating random numbers with size of N_m as array of start points for mutation, call it P_m ;
        $S_{synthetic}$[i]=`newSeq` (s,$HMMp_+$,P_m);
        $p_+ = P_e(S_{synthetic}[i], HMMp_+)$; (* emmitance probability of synthesized sequence by $HMMp_+$ *)
        $p_-[] = P_e(S_{synthetic}[i], HMMp_-[])$;
    until $p_+ < \max p_-[]$;
end
return $S_{synthetic}$

Function `newSeq`(s,$HMMp_+$,P_m)

Input : s, original sequence; $HMMp_+$, HMM profile of set S_+ to which s belongs; P_m, array of start points for mutation

Output: $s_{synthetic}$, synthetic sequence from s

$s_{synthetic} = s$;
for $i \leftarrow 1$ **to** $|P_m|$ **do**
    $p = P_m[i]$; (* assume that $HMMp_+$ in position p has emitted $s[p]$ *)
    repeat
        $s_{synthetic}[p+1]$= emitted residue in position $p+1$ by $HMMp_+$;
        $p = p + 1$;
    until *(newres* $\neq s[p]$) && *(*$p < |HMMp_+|$) ;
end
return $s_{synthetic}$

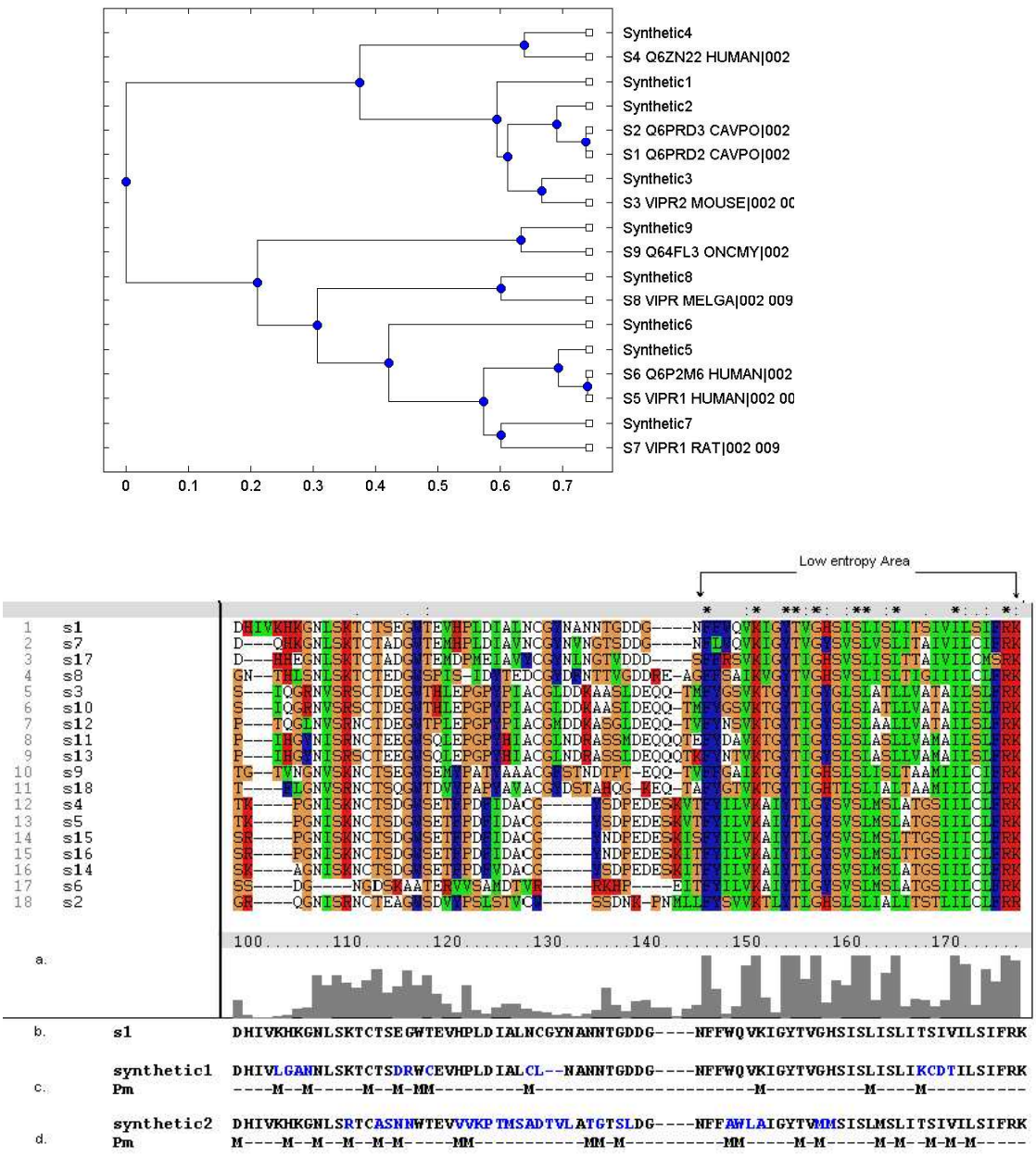

Fig. 1. The phylogenetic tree of the original and the synthesized sequences from the "vasoactive intestinal polypeptide" family of GPCRs (**upper**) and an example of the SPSO algorithm for sequences from the above family (**lower**). **a.** Multiple sequence alignment and low entropy area of that family **b.** A part of sequence $s1$. **c.** Synthetic sequence of $s1$ with N_m=50 . **d.** Synthetic sequence of $s1$ with N_m=100 (P_m: array of start points, shown by M, for mutations).

profile of the minor class and consider it as root node. Then, we suppose the root node has been mutated to synthesize all other sequences in the minor class through the *newSequence* procedure of our algorithm. It means each sequence is a mutated replica of the root node sequence which is emitted by the HMM profile of the minor class. We gain the value of N_m for each sequence. Then, we get the average of all those values as N_m entry for the SPSO algorithm.

With each call of the SPSO algorithm, we double the minor class. As an example of random synthesizing of sequences, Fig. 1(upper) shows the phylogenetic tree of the original sequences and the synthesized sequences for the vasoactive intestinal polypeptide family of class B (9 out of 18 sequences were randomly selected). It is shown that the synthesized sequences of most original sequences have less distance to them than to other sequences. In that figure (lower) we see two synthetic sequences of $s1$ with different values of N_m. In the low entropy area of the HMM profile of that family we have less mutations.

3 Datasets

To evaluate the performance of our algorithm, we ran our experiments on a series of both real and artificial datasets, whose specification covers different complexity and allows us to fully interpret the results. We want to check its efficiency with different ratio of imbalance and complexity. Fig. 3 shows the pictorial representation of our datasets. In the first one, the distribution of the positive and negative data are completely different and they are separate from each other. With that distribution, we want to see, how the imbalance ratio affects the performance of the classifier by itself. The second one shows datasets in which positive data are closer to negative data and there is an overlap between the minor and major classes. With this distribution, we can consider both the ratio of imbalance and overlap of the datasets in our study. The third one is a case where the minor class completely overlaps with the major class and we have fully overlapping data.

We used the G-protein coupled receptors (GPCRs) family as real data and then created artificial data based on it. G-protein coupled receptors (GPCRs) are a large superfamily of integral membrane proteins that transduce signals across the cell membrane [12]. Through their extracellular and transmembrane domains they respond to a variety of ligands, including neurotransmitters, hormones and odorants. They are characterized by seven hydrophobic regions that pass through the cell membrane (transmembrane regions), as shown in Fig. 2.

According to the binding of GPCRs to different ligand types they are classified into different families. Based on GPCRDB (G protein coupled receptor database) [13] all GPCRs have been divided into a hierarchy of 'class', 'subfamily', 'sub-sub-family' and 'type'. The dataset of this study was collected from GPCRDB and we used the latest dataset (June 2005 release). The six main families are: Class A (Rhodopsin like), Class B (Secretin like), Class C (Metabotropic glutamate/pheromone), Class D (Fungal pheromone), Class E (cAMP receptors) and Frizzled/Smoothened family. The sequences of proteins in GPCRDB were

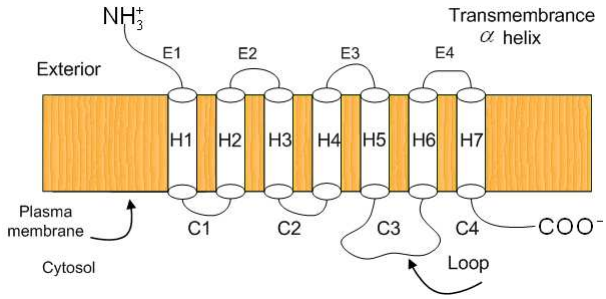

Fig. 2. Schematic representation of GPCR shown as seven transmembrane helices depicted as cylinders along with cytoplasmic and extracellular hydrophilic loops.

taken from SWISS-PROT and TrEMBL [14]. All six families of GPCRs (5300 protein sequences) are classified in 43 subfamilies and 99 sub-subfamilies.

If we want to classify GPCRs at the sub-subfamily level, mostly we have only a very low number of protein sequences as positive data (minor class) compared with others (major class). We chose different protein families from that level to cover all states of complexity and imbalance ratio discussed above (Fig. 3). In some experiments we made artificial data using those families and synthesized sequences from them (discussed later). We used numbers to show the level of family, subfamily and sub-subfamily. For example 001-001-002 means the sub-subfamily Adrenoceptors that belongs to subfamily of Amine (001-001) and class A (001).

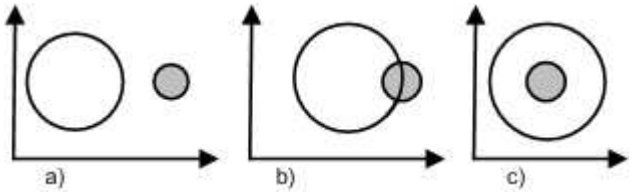

Fig. 3. Pictorial representation of the minor (shaded circle) and major classes of our datasets.

4 Experiments

We selected the peptide subfamily (001-002) of Class A (Rhodopsin-like) to classify its 32 families (or sub-subfamily level of class A). We built HMM profiles of all families and measured the probability of emission of sequences belonging to each one by all HMM profiles. We saw that the emission probability of each sequence generated by the HMM profile of its own family is higher than that of

almost all other families. So we can conclude that the distribution of the peptide subfamily in a suitable feature map can be considered as in Fig. 3.a. In this study, we used the local alignment kernel (LA kernel) [15] to generate vectors from protein sequences. It has been shown that the local alignment kernel has better performance than other previously suggested kernels for remote homology detection when applied to the standard SCOP test set [10]. It represents a modification of the Smith-Waterman score to incorporate sub alignments by computing the sum (instead of the maximum) over all possible alignments. We build a kernel matrix K for the training data. Each cell of the matrix is a local alignment kernel score between protein i and protein j. Then we normalize the kernel matrix via $K_{ij} \leftarrow K_{ij}/\sqrt{K_{ii}K_{jj}}$. Each family is considered as positive training data and all others as negative training data. After that the SVM algorithm with RBF kernel is used for training. For testing, we created feature vectors by calculating a local alignment kernel between the test sequence and all training data. The number of sequences in the peptide subfamily is in the range of 4 to 251 belonging to (001-002-024) and (001-002-008), respectively. Thus the *imbalance ratio* varies from $\frac{4}{4737}$ to $\frac{251}{4737}$. Fig. 4.**a** shows the result of SPSO oversampling for classification of some of those families. We see that this method can increase the accuracy and sensitivity of the classifier faced with highly imbalanced data without decreasing its specificity. The minority class was oversampled at 100%, 200%, 300%,..., 800% of its original size. We see that the more we increase the synthetic data (oversample) the better result we get, until we get the optimum value. It should be noted that after oversampling, the accuracy of classifiers for the major class didn't decrease.

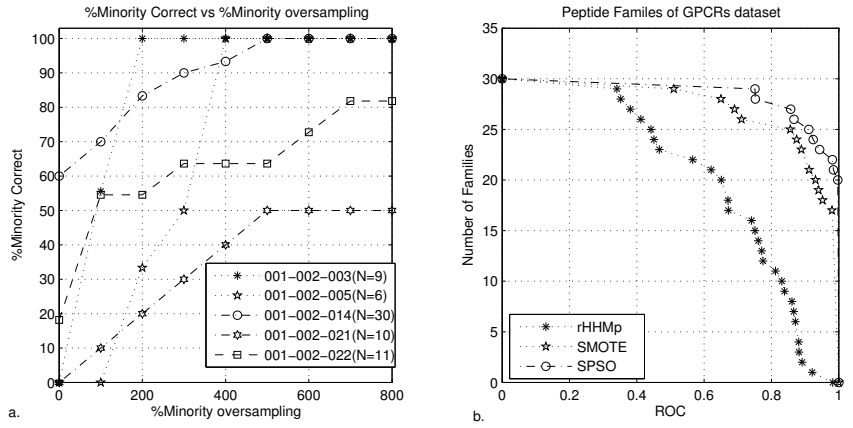

Fig. 4. a. %Minority correct for SPSO oversampling for some families of peptide subfamily (N number of sequences). **b**. Comparison of several methods for oversampling. The graph plots the total number of families for which a given method exceeds an ROC score threshold.

We compared our method with two other methods. The first one was SMOTE (Synthetic Minority Oversampling Techniques) [8] that operates in the feature space rather than in data space, so it works with all kind of data. The second comparison was done with randomly oversampling, in which we create random sequences by the HMM profile of each family. For this, like our method, we build a multiple alignment of the minor class sequences using ClustalW and then train a hidden Markov model profile with length of the created multiple alignment sequence. Then, we create random sequences by the HMM profile of each family. In this method we don't have enough control on the distribution of created random sequences. We call this method rHMMp in this paper.

In our study, we used the Bioinformatics Toolbox of MATLAB to create the HMM profiles of families and the SVMlight package [16], to perform SVM training and classification.

We used the Receiver Operating Characteristic (ROC) graphs [17] to show the quality of the SPSO oversampling technique. An ROC graph characterizes the performance of a binary classifier across all possible trade-off between the classifier sensitivity (TP_{rate}) and false positive error rates (FP_{rate}) [18]. The closer the ROC score is to 1, the better performance the classifier has. We oversampled each minority class with the three different methods noted above, until we got the optimum performance for one of them. At that point, we calculated the ROC score of all methods.

Fig. 4.**b** shows the quality of classifiers when using different oversampling methods. This graph plots the total number of families for which a given method exceeds an ROC score threshold. The curve of our method is above the curve of other methods and shows better performance. In our method and in SMOTE, the inserted positive examples have been created more accurately than random oversampling (rHMMp). Our method (SPSO) outperforms the other two methods especially for families in which we have a low number of sequences, although the quality of the SMOTE is comparable to the SPSO method.

To study the second and third representation of the dataset shown in Fig. 3 we had to create some sequences synthetically. At first we built the HMM profile of each family of the peptide families and then computed the probability score of each sequence when emitted not only by the HMM profile of its own family but also from all other families. The average of those scores for sequences of each family when emitted by each HMM profile can be used as a criterion for the closeness of the distribution of that family to other families and how much it can be represented by their HMM profiles. In this way we can find the nearest families to each peptide family. After that we synthesized sequences for each family through the *newSeq* procedure of the SPSO algorithm, provided that it is emitted by the HMM profile of another near family and not by its own HMM profile. So after each start position for mutation (Fig.1 (lower)) we have residues that are emitted by another HMM profile (we want to have overlap with) instead of its own HMM profile and there is an overlap for the distribution of synthesized sequences between those two families. The degree of overlapping can be tuned by the value of N_m (number of mutations). This dataset (original and

new synthesized sequences) can be considered as partially overlapping dataset (Fig. **3.b**). If we create more sequences using other HMM profiles the distribution of the dataset is fully overlapping (Fig. **3.c**). To study the partially overlapping datasets, we selected 10 families of peptide families and built the synthesized sequences as noted above. To create the fully overlapping dataset, we performed that routine for each family using the HMM profile of three families near to the original family, separately.

We compared our oversampling technique with the SMOTE oversampling technique and the different error cost (DEC) method [6]. Tables 1 and 2 show the results. We see that in general SPSO outperforms the SMOTE and DEC methods, and the performance of the classifier with the SPSO oversampling technique in fully overlapped datasets is more apparent. When there is more overlapping between the minor and major classes, the problem of imbalanced data is more acute. So the position of the inserted data in the minor class is more important and in our algorithm it has been done more accurately than in SMOTE method. With regard to the time needed for each algorithm, DEC has an advantage compared to our method, because in the oversampling technique the minor class, depending on the number of its instances, is oversampled up to 10 times (in our experiments) which increases the dimension of the of kernel matrix. In contrast, in the DEC method choosing the correct cost of error for minority and majority classes is an important issue. One suggested method is to set the error cost ratio equal to the inverse of the imbalance ratio. But, based on our experiments that value is not always the optimum, and especially in partially and fully overlapped datasets we had instability of performance even with values near the optimal value. Based on our experiments in the well-separated imbalanced data the quality of DEC is very near to the SPSO method and for some experiments, even better, and we could find optimum value for error cost ratio simply. So perhaps with this kind of datasets one should prefer the DEC method. But with partially and fully overlapping data, we found that our oversampling method in general has better performance, and if it is used along with the DEC method, it not only increases the performance of the classifier but it also makes finding the value for the error cost ratio simpler. We also have more stability with values close to the optimum value of the error cost ratio. The graphs in Fig. 5.**a** and Fig. 5.**b** show the value of the ROC score of classifier for partially overlapped artificial sequences from the family of 001-002-024 ($001 - 002 - 024^{'}$) when the DEC method and DEC along with SPSO (400% oversampling) were applied. We see that when SPSO oversampling is used we have stability in ROC score values and after the optimum value, the ROC score does not change. The drawback is, that we again have to find the best value for the error cost ratio and the rate of oversampling through the experiment by checking different values, but in less time compared to only the DEC method because of the stability that was shown in Fig. 5.**b**. We used that method for all partially and fully overlapping artificial data (Table 1 and 2). For each experiment we oversampled data in different rates and selected different values of error cost ratio until we got the best result. The results in Fig. 5.**c**

show that for those kind of data the ROC scores of SPSO and DEC + SPSO are nearly the same. But in the second method (DEC + SPSO), we needed to oversample data less than in SPSO only method and we could find the best value of the error cost ratio sooner than in DEC only. With less rate of oversampling in SPSO we get less accurate results but we can compensate that with DEC.

Partially overlapping classes- ROC scores

minority class	# of sequences	SMOTE	DEC	SPSO
$001-002-015^{'}$	16	0.863	0.943	0.951
$001-002-016^{'}$	122	0.821	0.912	0.929
$001-002-017^{'}$	68	0.854	0.892	0.884
$001-002-018^{'}$	74	0.912	0.871	0.891
$001-002-020^{'}$	86	0.972	0.975	0.984
$001-002-021^{'}$	40	0.695	0.739	0.723
$001-002-022^{'}$	44	0.725	0.762	0.751
$001-002-023^{'}$	48	0.965	0.982	0.996
$001-002-024^{'}$	8	0.845	0.834	0.865
$001-002-025^{'}$	10	0.945	0.972	0.987
overall ROC-score		**0.859**	**0.882**	**0.896**

Table 1. ROC scores obtained on the partially overlapping classes created from peptide families of GPCR dataset, by various methods. DEC = different error cost;

Fully overlapping classes- ROC scores

minority class	# of sequences	SMOTE	DEC	SPSO
$001-002-015^{"}$	32	0.673	0.680	0.724
$001-002-016^{"}$	244	0.753	0.775	0.821
$001-002-017^{"}$	136	0.672	0.652	0.643
$001-002-018^{"}$	148	0.591	0.624	0.672
$001-002-020^{"}$	172	0.763	0.821	0.858
$001-002-021^{"}$	80	0.632	0.689	0.681
$001-002-022^{"}$	88	0.615	0.812	0.854
$001-002-023^{"}$	96	0.912	0.942	0.968
$001-002-024^{"}$	16	0.716	0.768	0.819
$001-002-025^{"}$	20	0.908	0.902	0.921
overall ROC-score		**0.723**	**0.766**	**0.796**

Table 2. ROC scores obtained on the Fully overlapping classes created from peptide families of GPCR dataset by various methods.

For further evaluation of our method, we used our oversampling technique in classification all GPCRs families at sub family and sub-sub family level (mostly we have low number of sequences). In subfamily classification we randomly parti-

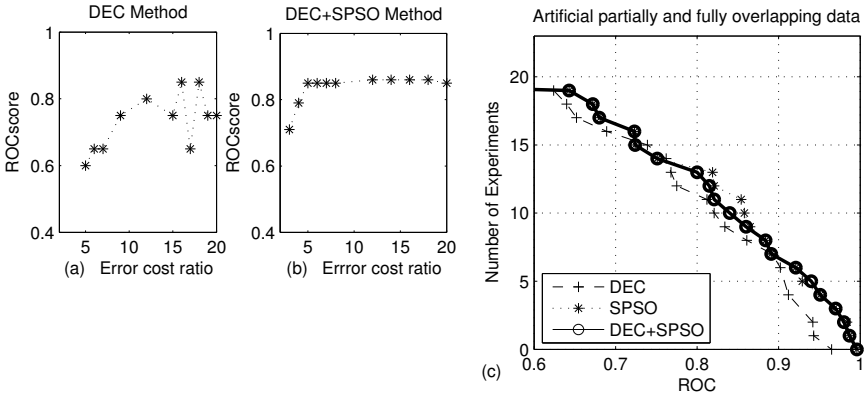

Fig. 5. (**a**) Comparison of the ROC score at different error cost ratios for artificial sequences of $001-002-024^{'}$ in classifier with DEC method. (**b**) classifier with DEC + SPSO mothods (400% oversampled). (**c**). Comparison of DEC, SPSO and DEC+SPSO methods for imbalanced data. The graph plots the total number of experiments of partially and fully overlapped imbalanced artificial data for which a given method exceeds an ROC score threshold.

tioned the data in two non-overlapping sets and used a two-fold cross validation protocol. The training and testing was carried out twice using one set for training and the other one for testing. To compare with the results of other researchers, the prediction quality was evaluated by Accuracy (ACC), Matthew's correlation coefficient (MCC), overall Accuracy ($\overline{ACC}$)and overall MCC ($\overline{MCC}$) as follows:

$$ACC = \frac{TP + TN}{(TN + FN + TP + FP)} \tag{1}$$

$$MCC = \frac{TP \times TN - FN \times FP}{\sqrt{(TN + FN)(TP + FN)(TN + FP)(TP + FP)}} \tag{2}$$

$$\overline{ACC} = \sum_{i=1}^{N} \frac{ACC(i)}{N} \tag{3}$$

$$\overline{MCC} = \sum_{i=1}^{N} \frac{MCC(i)}{N} \tag{4}$$

(TP = true positive, TN = true negative, FP = false positive , FN = false negative, N=number of subfamily or sub-subfamily)

The overall accuracy we gained for families A, B and C is 98.94%, 99.94% and 96.95%, respectively, and overall MCC for families A, B and C is 0.98, 0.99 and 0.91, respectively. Almost all of the subfamilies are accurately predicted with our method. Tables 3 shows the results of subfamily classification for classes A of

GPCRs. At the subfamily level we compared our method with that of Bhasin et al. [19]. They used an SVM-based method with dipeptide composition of protein sequences as input. The accuracy and MCC values of our method outperform theirs. For example in classification of subfamily A, the overall accuracy and MCC of their method were 97.3% and 0.97 but ours are 98.4% and .98, respectively. They did a comparison with other previously published methods like that of Karchin et al. [20] and showed that their method outperformed the others.

Table 3. The performance of our method in GPCRs subfamily classification(Class A).

Class A subfamilies	Accuracy (%)	MCC
Amine	99.9	0.99
Peptide	97.8	0.97
Hormone protein	100.0	1.00
(Rhod)opsin	99.6	0.99
Olfactory	99.9	0.99
Prostanoid	99.9	.98
Nucleotide-like	100.0	1.00
Cannabinoid	100.0	1.00
Platelet activating factor	100.0	1.00
Gonadotropin-releasing hormone	100.0	1.00
Thyrotropin-releasing hormone	100.0	1.00
Melatonin	100.0	1.00
Viral	87.0	0.8
Lysosphingolipid	100.0	1.00
Leukotriene	100.0	1.00
Overall	98.4	0.98

For sub-subfamily classification we used 5-fold cross validation. Table 4 shows the results for the sub-subfamily level. We see that in this level also the accuracy is high and we could classify most of GPCRs sub-subfamilies. We could obtain an overall accuracy of 97.93% and a MCC of 0.95 for all sub-subfamilies. At this level we could increase the accuracy, especially when the number of sequences in the positive training data was less than 10, and there was no example in which with our oversampling method the accuracy decreases.

To the best of our knowledge there is only one study which has been done for sub-subfamily classification [21] in GPCRs families. Their approach is based on bagging a classification tree and they achieved 82.4% accuracy for sub-subfamily classification, which is less accurate than ours (97.93% with MCC of 0.95) despite the fact that they had excluded families with less than 10 sequences (we only excluded families with less than 4 sequences). We think our oversampling technique can be widely used for other applications of protein classification with the problem of imbalanced data and it can be used along with the different error cost (DEC) method to overcome the problem of imbalanced data for protein data.

Table 4. The performance of our method in GPCRs sub-subfamily classification for Class A,B and C.

Class A	subfamilies	Overall Accuracy (%)	Overall MCC
	Amine	97.1	0.91
	Peptide	99.9	0.93
	Hormone protein	100.1	1.00
	(Rhod)opsin	96.6	0.95
	Olfactory	98.9	0.92
	Prostanoid	98.0	0.94
	Gonadotropin-releasing hormone	96.1	0.93
	Thyrotropin-releasing hormone	91.2	0.94
	Lysosphingolipid	98.4	1.00
Class B	Latrophilin	100.0	1.00
Class C	Metabotropic glutamate	98.1	0.96
	Calcium-sensing like	97.2	0.93
	GABA-B	100.0	1.00
Overall		97.93	0.95

5 Conclusion

In this work, we suggested a new approach of oversampling for the imbalanced protein data in which the minority class in the data space is oversampled by creating synthetic protein sequences, considering the distribution of the minor and major classes. This method can be used for protein classification problems and remote homology detection, where classifiers must detect a remote relation between unknown sequence and training data with an imbalance problem. We think that this kind of oversampling in kernel-based classifiers not only pushes the class separating hyperplane away from the positive data to negative data but also changes the orientation of the hyperplane in a way that increases the accuracy of classifier. We developed a systematic study using a set of real and artificially generated datasets to show the efficiency of our method and how the degree of class overlapping can affect class imbalance. The results show that our SPSO algorithm outperforms other oversampling techniques. In this paper, we also presented evidences suggesting that our oversampling technique can be used along with DEC to increase its sensitivity and stability. For further work, we hope to find an algorithm for finding the suitable rate of oversampling and error cost ratio when DEC and SPSO methods are used together.

References

1. C.Leslie, E. Eskin, A. Cohen, J. Weston, and W.S. Noble. Mismatch string kernel for svm protein classification. *Advances in Neural Information Processing System*, pages 1441–1448, 2003.
2. A. Al-Shahib, R. Breitling, and D. Gilbert D. Feature selection and the class imbalance problem in predicting protein function from sequence. *Appl Bioinformatics*, 4(3):195–203, 2005.

3. M. Pazzini, C. Marz, P. Murphi, K. Ali, T.Hume, and C. Bruk. Reducing misclassification costs. *In proceedings of the Eleventh Int. Conf. on Machine Learning*, pages 217–225, 1994.
4. N. Japkowicz, C.Myers, and M. Gluch. A novelty detection approach to classification. *In Proceeding of the Fourteenth Int. Joint Conf. on Artificial Inteligence*, pages 10–15, 1995.
5. N. Japkowicz. Learning from imbalanved data sets: A comparison of various strategies. *In Proceedings of Learning from Imbalanced Data*, pages 10–15, 2000.
6. K. Veropoulos, C. Campbell, and N. Cristianini. Controlling the sensitivity of support vector machines. *Proceedings of the International Joint Conference on AI*, pages 55–60, 1999.
7. G. Wu and E. Chang. Class-boundary alignment for imbalanced dataset learning. *In ICML 2003 Workshop on Learning from Imbalanced Data Sets II, Washington, DC*, 2003.
8. Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence and Research*, 16:321–357, 2002.
9. C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. *Proceedings of the Pacific Symposium on Biocomputing*, page 564575, 2002.
10. H. saigo, J. P. Vert, N. Ueda, and T. akustu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.
11. J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustalw: improving the sesitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22:4673–4680, 1994.
12. T. K Attwood, M. D. R. Croning, and A. Gaulton. Deriving structural and functional insights from a ligand-based hierarchical classification of g-protein coupled receptors. *Protein Eng.*, 15:7–12, 2002.
13. F. Horn, E. Bettler, L. Oliveira, F. Campagne, F. E. Cohhen, and G. Vriend. Gpcrdb information system for g protein-coupled receptors. *Nucleic Acids Res.*, 31(1):294–297, 2003.
14. A. Bairoch and R. Apweiler. The swiss-prot protein sequence data bank and its supplement trembl. *Nucleic Acids Res.*, 29:346–349, 2001.
15. J.-P.Vert, H. Saigo, and T.Akustu. *Convolution and local alignment kernel In B. Schoelkopf, K. Tsuda, and J.-P.Vert (Eds.), Kernel Methods in Compuatational Biology.* The MIT Press.
16. T. Joachims. Macking large scale svm learning practical. *Technical Report LS8-24*, Universitat Dortmond, 1998.
17. F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 423:203–231, 2001.
18. J. Swet. Measuring the accuracy of diagnostic systems. *Science.*, 240:1285–1293, 1988.
19. M. Bhasin and G. P. S. Raghava. Gpcrpred: an svm-based method for prediction of families and subfamilies of g-protein coupled receptors. *Nucleaic Acids res.*, 32:383–389, 2004.
20. R. Karchin, K. Karplus, and D. Haussler. Classifying g-protein coupled receptors with support vector machines. *Bioinformatics*, 18(1):147159, 2002.
21. Y. Huang, J. Cai, and Y. D. Li. Classifying g-protein coupled receptors with bagging classification tree. *Computationa Biology and Chemistry*, 28:275–280, 2004.