ProWSyn: Proximity Weighted Synthetic Oversampling Technique for Imbalanced Data Set Learning

Sukarna Barua[1], Md. Monirul Islam[1], and Kazuyuki Murase[2]

[1] Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh
[2] University of Fukui, Fukui, Japan

Abstract. An imbalanced data set creates severe problems for the classifier as number of samples of one class (majority) is much higher than the other class (minority). Synthetic oversampling methods address this problem by generating new synthetic minority class samples. To distribute the synthetic samples effectively, recent approaches create weight values for original minority samples based on their importance and distribute synthetic samples according to weight values. However, most of the existing algorithms create inappropriate weights and in many cases, they cannot generate the required weight values for the minority samples. This results in a poor distribution of generated synthetic samples. In this respect, this paper presents a new synthetic oversampling algorithm, Proximity Weighted Synthetic Oversampling Technique (ProWSyn). Our proposed algorithm generate effective weight values for the minority data samples based on sample's proximity information, i.e., distance from boundary which results in a proper distribution of generated synthetic samples across the minority data set. Simulation results on some real world datasets shows the effectiveness of the proposed method showing improvements in various assessment metrics such as AUC, F-measure, and G-mean.

Keywords: Imbalanced learning, clustering, synthetic oversampling.

1 Introduction

Imbalanced learning problem is to deal with imbalanced data sets where one or more classes have much higher number of data samples than other classes. Generally speaking, imbalanced learning occurs whenever some types of data distribution significantly dominate the sample space compared to other data distributions. By convention, in imbalanced data sets, we call the classes having more samples the majority classes and the ones having fewer samples the minority classes. Classifiers tend to produce greater classification errors over the minority class samples [1–3]. Many real world problems suffer from this phenomenon such as medical diagnosis [4], information retrieval [5], detection of fraudulent telephone calls [6] and oil spills in radar images [7], data mining from direct marketing [8], and helicopter fault monitoring [9]. Thus the identification of the minority class samples is of utmost importance.

There have been many attempts in solving imbalanced learning problems, such as various oversampling and undersampling methods [10]. Roughly speaking, an undersampling method remove some majority class samples from an imbalanced data set with an aim to balance the distribution between the majority class and minority class samples [11–14], while an oversampling method does the opposite, i.e, generates synthetic minority class samples and adds them to the data set [15–19]. Both undersampling and oversampling methods have been shown to improve classifiers' performance on imbalanced data sets. While comparing oversampling and undersampling one natural observation favoring oversampling is that undersampling may remove essential information from the original data, while oversampling does not suffer from this problem. It has been shown that oversampling is lot more useful than undersampling and the performance of oversampling was shown to improve dramatically even for complex data sets [20].

Most of the existing oversampling methods (e.g. [17–19]) first try to estimate the difficulty levels of the minority class samples in generating synthetic minority class samples. In doing so, they assign weights to the minority class samples based on a computed value, i.e., δ. The methods then use δ to decide how many synthetic samples are to be generated for a particular minority class sample. In Sect. 2, we illustrate that the way δ is computed is not reasonable in many scenarios. Consequently, the sample generation process is not able to generate synthetic samples appropriately, which affects classifiers performance.

In this paper, we present a new flexible oversampling method, named Proximity Weighted Synthetic Oversampling Technique (ProWSyn). Unlike previous work, ProWSyn uses the distance information of the minority class samples from the majority samples in assigning weights to the minority class samples. The effectiveness of the proposed technique has been evaluated on several benchmark classification problems with high imbalanced ratio. It has been found that ProWSyn performs better compared to some other existing methods in most of the cases.

The remainder of this paper is divided into five sections. In Sect. 2, we present the related works for solving imbalanced learning problems. Section 3 describes the details of the proposed algorithm. In Sect. 4, we present the experimental study and simulation results. Finally, in Sect. 5, we provide some future aspects of this research and conclude the paper.

2 Related Work

The main objective of oversampling methods is to oversample the minority class samples to shift the classifier learning bias toward the minority class. Synthetic Minority Over-sampling Technique (SMOTE) is one such method [15]. For every minority class sample, this technique first finds its k (which is set to 5 in SMOTE) nearest neighbors of the same class and then randomly selects some of them according to the over-sampling rate. Finally, SMOTE generates new synthetic samples along the line between the minority sample and its selected nearest neighbors. The problem of SMOTE is that it does not consider the importance of minority class samples, thus generates an arbitrary equal number of synthetic minority class samples. However, all the minority class samples are not equally important (hard). The minority class samples that are surrounded by many majority class samples or closer to the classifier's decision boundary are more important than the other ones.

Adaptive synthetic (ADASYN) oversampling [17], Ranked Minority Oversampling (RAMO) [18] and CBSO [19] try to address the aforementioned problem in dealing with imbalanced learning problems. These methods are based on the idea of assigning weights to minority class samples according to their importance. These weights are used for generating synthetic samples. More synthetic samples are generated for a large weight than for a small weight. CBSO, like earlier approaches [17, 18], uses a parameter δ, the number of majority samples among the k nearest neighbors of a minority sample x, for assigning weight to x. Let , $N_k(x)$ is the set of k nearest neighbors of x. Then, δ for x equals to the number of the majority class samples in $N_k(x)$. If this number is large, then δ is high, resulting a large weight assignment to the minority sample [17–19]. However, the use of δ for assigning weights to individual minority samples may not be appropriate in the situations described below.

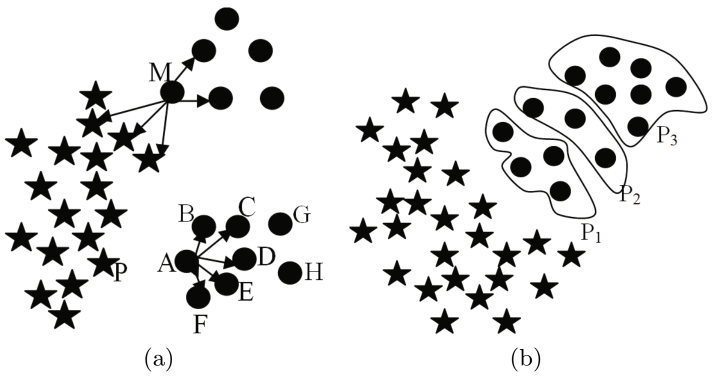

Fig. 1. (a) k nearest neighbors are shown by arrows for some minority samples. (b) Minority samples are partitioned according to their proximity, i.e., distance from boundary.

1. *δ may be inappropriate for assigning weights to the minority class samples located near the decision boundary.*
   To understand this fact, consider Fig. 1(a) where the minority class and majority class samples are shown by circles and stars respectively. It is clear from this figure that the minority class sample A has no majority class samples in their k-nearest neighborhood (assume $k = 5$) and $N_5(A)$ contains only the minority class samples B, C, D, E and F. We use arrow in Fig. 1(a) to show the neighbors of a minority sample. Since, $N_k(A)$ does not contain

any majority class sample, δ for A would be 0. For similar reasons, δ for B would also be 0. It means A and B will be given zero or the lowest weight, although seemingly they are the most important samples due to their position near the decision boundary.

2. *δ may be insufficient to discover the difference of minority samples w.r.t their importance in learning.* It can be seen from Fig. 1(a) that the minority samples A and B are closer to the decision boundary than those of G and H. It is, therefore, reasonable that A and B should be given higher weight than G and H. However, if we assume $k = 5$ and compute δ for A, B, G and H, then it is evident from the figure that the δ would is 0 for each of them, because no $N_5(x)$ ($x \in \{A, B, G, H\}$ contain any majority class sample. It is now understood that δ cannot differentiate the minority class samples according to their importance in learning. Another problem is that δ for some or all samples, i.e. A, B and so on, of one region, is 0, while for the sample M of another region, δ gets a good positive value (Fig. 1(a)). Under this condition, all synthetic samples will be generated in the M's region and no or very few will be generated in the A's region. This example again illustrate the same thing i.e. δ cannot discover the difference of the minority class samples w.r.t their importance in learning.

The above scenarios confirms that earlier approaches based on δ e.g. [17–19] cannot effectively assign weights to the minority class samples. In most scenarios, if the parameter k is not sufficiently large, then most of the minority samples will get zero weight [17, 19] or the lowest weight [18]. Minority class samples having zero or the lowest weight will get no or very few synthetic samples around them. It may seem the problem can be avoided by increasing the value of k to contain the majority class samples. However, the required value of k cannot be determined in advance. In some regions, a small k may suffice, while in other regions a large k is to be required. Even if k is increased, it cannot solve the skewed distribution of synthetic samples. For example, suppose we increase k to 6 to contain the majority class samples for the minority class sample A (Fig. 1(a)). For this case, $N_6(A) = \{B, C, D, E, F, P\}$, which contains one majority sample P. Hence, A's delta will be 1. However, M's delta will be 4, because $N_6(M)$ contains four majority class samples (Fig. 1(a)). Still, A's δ is smaller compared to that for M and more synthetic samples will be generated in the neighborhood of M. This justifies that increasing k cannot effectively solve the problem at all.

3 Proposed Algorithm

Motivated by problems stated in Sect. 2, we have extended the recently proposed CBSO [19] algorithm and proposed a new improved algorithm, named Proximity Weighted Synthetic Oversampling Technique (ProWSyn). The new algorithm uses a different weight generation technique to alleviate the problems described earlier. The complete algorithm is shown in [Algorithm ProWSyn]. Our ProWSyn differs from CBSO in Steps 2 to 7 in which each minority sample

x is now weighted based on the proximity level of x i.e.,PL_x. We measure PL_x using the Euclidean distance of x from the majority class samples. Steps 8 to 11 create the synthetic samples and produce an output oversampled minority data set, S_{omin}. The details of weight generation procedure are discussed below.

[**Algorithm ProWSyn**]
Input:
Training data samples D_{tr} with m samples $\{x_i, y_i\}$, $i = 1 \cdots m$, where x_i is an instance in n dimensional feature space X, and $y_i \in \{-1, 1\}$ is the class identity level associated with x_i. Define S_{maj} and S_{min} to be the majority and minority class set respectively and m_l and m_s as the number of majority class and minority class samples respectively. Therefore, $m_s \leq m_l$ and $m_s + m_l = m$.
Procedure:

1. Calculate the number of synthetic samples that need to be generated for the minority class:
$$G = (m_l - m_s) \times \beta$$
where $\beta \in [0, 1]$ is a parameter used to specify the desired balance level after generation of the synthetic samples. We can obtain a fully balanced dataset by assigning $\beta = 1$.
2. Initialize, $P = S_{min}$
3. For $i = 1$ to $L - 1$ do the following:
   (a) From each majority sample y, find the nearest K minority samples in P according to Euclidean distance. Let, the set of these K samples to be $N_K(y)$.
   (b) Form partition P_i as the union of all $N_K(y)$s:
$$P_i = \bigcup_{y \in S_{maj}} N_K(y) \tag{1}$$
   (c) Set proximity level of each minority sample x in partition P_i to be i:
$$PL_x = i, \forall x \in P_i \tag{2}$$
   (d) Remove selected minority samples from P, $P = P - P_i$.
4. Form partition P_L with the remaining unpartitioned samples in P.
5. Set proximity level of each x in P_L to be L:
$$PL_x = L, \forall x \in P_L$$
6. For each x, calculate a weight w_x from its proximity level PL_x defined as:
$$w_x = \exp\left(-\theta * (PL_x - 1)\right) \tag{3}$$
where θ is a smoothing factor and $w_x \in [0, 1]$.
7. Normalize w_x according to $\widehat{w_x} = w_x / \sum_{z \in S_{min}} w_z$, so that $\widehat{w_x}$ is a density distribution ($\sum \widehat{w_x} = 1$)

```
 8. Calculate the number of synthetic samples $g_x$ that need to be generated for $x$:
        $g_x = \widehat{w_x} \times G$
 9. Find the clusters of minority set, $S_{min}$
10. Initialize set, $S_{omin} = S_{min}$
11. For each $x$, generate $g_x$ synthetic minority class samples according to the
    following steps:
    Do the loop from 1 to $g_x$
      (a) Randomly select one minority sample $y$, from $x$'s cluster (as found in
          step 9).
      (b) Generate a synthetic sample, $s$, according to
          $s = x + \alpha \times (y - x)$, where $\alpha$ is a random number in the range $[0, 1]$.
      (c) Add $s$ to $S_{omin}$: $S_{omin} = S_{omin} \bigcup \{s\}$
    End Loop
```

Output: Oversampled minority data set, S_{omin}

3.1 Weight Generation Mechanism of ProWSyn

The goal of our weight generation mechanism is assign appropriate weights to the minority class samples according to their importance in learning. To do this, ProWSyn works in two phases. In first phase (Steps 2-5), it divides the minority data set in a number of partitions (say, L) based on their distance from the decision boundary. Each partition is assigned a proximity level where the level increases with increasing distance from the boundary. Minority class samples with lower proximity levels are the difficult samples and therefore are important for learning, while they with higher proximity levels are less important and may not have significant importance at all. In second phase, ProWSyn generates synthetic minority class samples using the proximity information so that it can generate more synthetic samples in lower proximity regions, i.e., regions that are very near to the decision boundary. The whole procedure is described below (Steps 2 to 7 of [Algorithm ProWSyn]).

From each majority sample ProWSyn finds the nearest K minority class samples according to the Euclidean distance. The set of all these minority class samples form the first partition, P_1 (of proximity level 1), the nearest level of samples from the boundary. Then, it finds the next K minority samples according to distance from each majority. These samples together form the second partition, P_2 (of proximity level 2). In this way, the procedure is repeated for $L-1$ such partitions of proximity levels 1 to $L-1$. The rest of the unpartitioned samples will form partition L, the farthest set of samples from the boundary. A simulated partitioning is shown in Fig. 1(b) for $K = 3$ and $L = 3$. It is seen that the minority class samples are properly identified and partitioned according to their distance from the boundary, which also signifies their importance in oversampling.

Table 1. Description of data set characteristics used in simulation

Dataset	Minority Class	Features	Instances	Minority	Majority	%Minority
Pageblocks	Class of 'Graphic', 'Vert.line', 'Picture'	10	5476	231	5245	4%
Abalone	Class of '18'	7	731	42	689	6%
CTG	Class of '3', '4	21	2126	134	1992	6.3%
Segment	Class of 'Grass'	19	2310	330	1980	14%
Libra	Class of '1', '2', '3'	90	360	72	288	20%
Yeast	Class of ME3', 'ME2', 'EXC', 'VAC', 'POX', 'ERL'	8	1484	304	1180	21%
Robot	Class of 'slight-left-turn', 'slight-right-turn'	24	5456	1154	4302	22%
Vehicle	Class of '1'	18	940	219	721	24%
Breast-tissue	Class of 'CAR', 'FAD'	9	106	36	70	34%
Pima	Class of '1'	8	768	268	500	35%

Minority class samples are given weight according to their proximity level (Step 6 of [Algorithm ProWSyn]). All minority samples in the same partition, i.e., having same proximity level, gets the same weight. As the level increases, weight of the minority samples also decreases exponentially (Eqn. 3). The parameter θ in (3) controls the rate with which weight decreases with respect to levels.

The above weight generation technique of ProWSyn has several advantages over earlier δ-based approaches. First, our ProWSyn can effectively find weight for a minority sample according to its position from the decision boundary, while earlier approaches fail to do so (Sect. 2). Secondly, the proposed method successfully partitions the minority class samples based on the distance from the decision boundary. So, samples closer to boundary get higher weight than samples that are further. This is not guaranteed in earlier approaches (Sect. 2). Thirdly, while earlier approaches may lead to generation of synthetic samples in a few small regions due to positive weights of a few minority class samples (Sect. 2), ProWSyn can avoid it by assigning proper weight values for all minority class samples. Fourthly, the procedure of ProWSyn is a more general one, we can easily control the size of each partition and number of partitions to be created based on the problem domain by varying the parameters K and L. However, there is no such scope in earlier approaches.

4 Experimental Study

In this section, we evaluate the effectiveness of our proposed ProWSyn algorithm and compare its performance with ADASYN [17], RAMO [18], and CBSO [19] methods. We use two different classifier models: backpropagation neural network and C4.5 decision tree [21]. We collect ten datasets from UCI machine learning repository [22]. The data sets were chosen in such a way that they contained a varied level of imbalanced distribution of samples. Some of these original data sets were multi-class data. Since, we are only interested in two-class classification problem, these data sets were transformed to form two-class data sets in a way which ensures a certain level of imbalance. Table 1 shows the minority class composition (these classes in the original dataset were combined to form the minority class and rest of the classes form the majority class) and other characteristics of the data sets such as the number of features, the number of total samples, and the number of majority and minority class samples. As evaluation metrics, we use the most popular measure for imbalanced problem domains i.e., the area under the Receiver Operating Characteristics (ROC) graph [23], usually known as AUC. Furthermore, we use two other popular performance metrics such as F-measure and G-mean [10].

We run single decision tree classifier and single neural network classifier on the selected datasets described in Table 1. For the neural network classifier, the number of hidden neurons is randomly set to 5, the number of input neurons is set to be equal to the number of features in the dataset and the number of output neurons is set to 2. We use Sigmoid function as an activation function for neural network. The number of training epochs is randomly set to 300 and learning rate is set to 0.1. For ADASYN and CBSO, the value of the nearest neighbors, K, is set to 5 [17, 19]. The values of the nearest neighbors, i.e., $k1$ and $k2$, of RAMO are chosen as 5 and 10 respectively [18]. The scaling coefficient, α, for RAMO is set to 0.3 [18]. For ProWSyn and CBSO, $C_p = 3$ [17, 19]. For ProWSyn, other parameter values are $\theta = 1$, $K = 5$, and $L = 5$. These values are chosen after some preliminary simulation runs and they are not meant to be optimal. Number of synthetic samples generated is set by $\beta = 1$ for ADASYN, ProWSyn, and CBSO. Same number of synthetic samples were generated for RAMO for fair comparison.

Simulation results of F-measure (F-meas), G-mean, averaged AUC, and standard deviation of AUC values (SDAUC) are presented in Table 2. Each result is found after a 10-fold cross-validation. The best result in each category is highlighted with a bold-face type. We obtain averaged AUC values by averaging the AUC values of multiple simulation runs [18]. We also provide the number of times an algorithm win (win time) against any other method we compare here.

It is observed from Table 2 that for both classifiers, ProWSyn outperforms CBSO, ADASYN, and RAMO algorithms in terms of F-measure, G-mean, and averaged AUC in most of the datasets. As described in Sect. 2, δ based weight generation technique cannot create appropriate distribution of synthetic minority class samples. Generation of appropriate weight values by the ProWSyn approach leads to better distribution of synthetic samples along the difficult minority class regions making its performance better than other approaches. We apply Wilcoxon signed-rank test [24] to statistically compare the performance (AUC metric) of the four methods. The test is applied to compare our proposed

Table 2. Performance of PROWSYN, ADASYN [17], RAMO [18], and CBSO [19] on ten Real World Datasets using single Decision Tree and single Neural Network classifiers

		Decision Tree Classifier				Neural Network Classifier			
Dataset	Method	F-meas	G-mean	AUC	SDAUC	F-meas	G-mean	AUC	SDAUC
Pageblocks	ADASYN	0.9848	0.991	0.9833	0.0118	0.8692	0.9567	0.9723	0.0251
	RAMO	0.9864	0.9928	0.9854	0.011	0.9536	0.9827	0.9848	0.008
	CBSO	0.9834	0.9908	0.984	0.009	0.8932	0.9731	0.9828	0.0077
	PROWSYN	**0.9878**	**0.9939**	**0.9857**	0.0075	**0.9779**	**0.9911**	**0.9867**	0.0027
Abalone	ADASYN	0.3055	**0.5372**	0.7495	0.1919	0.4646	**0.6321**	0.8875	0.0609
	RAMO	0.2401	0.4361	0.7301	0.2013	0.393	0.5229	0.8892	0.0741
	CBSO	**0.3101**	0.5155	**0.7615**	0.2087	0.4493	0.5415	0.8938	0.0736
	PROWSYN	0.294	0.4772	0.7528	0.1706	**0.4728**	0.5965	**0.8976**	0.0706
CTG	ADASYN	0.6415	0.8093	0.891	0.0324	0.4637	0.6866	0.9016	0.0248
	RAMO	0.6639	0.8128	**0.9206**	0.0259	0.5179	0.7268	0.9125	0.0386
	CBSO	**0.6758**	**0.8250**	0.9067	0.0359	0.5434	**0.7514**	**0.9192**	0.0232
	PROWSYN	0.5882	0.7852	0.9137	0.0365	**0.5527**	0.73	0.914	0.0403
Segment	ADASYN	0.7227	0.8702	0.9609	0.0153	0.5258	0.7443	0.9062	0.0892
	RAMO	**0.7761**	**0.9068**	0.9653	0.0139	0.5792	0.7488	0.9391	0.0252
	CBSO	0.7509	0.8877	0.9547	0.015	0.5237	0.716	0.9164	0.0696
	PROWSYN	0.7152	0.8853	**0.9665**	0.0177	**0.6000**	**0.8018**	**0.9410**	0.027
Libra	ADASYN	0.7367	0.8347	0.8546	0.0861	0.8792	0.9153	0.8919	0.1557
	RAMO	0.7285	0.812	0.8327	0.1156	**0.9072**	**0.9338**	**0.9596**	0.0507
	CBSO	0.7278	0.8162	0.8397	0.1218	0.8832	0.931	0.9112	0.1527
	PROWSYN	**0.7717**	**0.8557**	**0.9054**	0.0707	0.8928	0.9284	0.9467	0.0953
Yeast	ADASYN	0.6224	0.7617	0.8728	0.0185	0.6693	0.8311	0.8906	0.0208
	RAMO	0.6381	0.7739	0.8766	0.0203	0.6837	0.8206	0.8825	0.022
	CBSO	0.6412	0.7726	0.8712	0.0267	**0.6936**	**0.8447**	0.8937	0.0176
	PROWSYN	**0.6631**	**0.7986**	**0.8865**	0.028	0.6821	0.8289	**0.8991**	0.0188
Robot	ADASYN	0.9832	0.9922	0.9863	0.0029	0.5702	0.7628	0.8189	0.0571
	RAMO	0.9819	0.9903	0.984	0.0027	0.5899	0.7692	0.8403	0.0466
	CBSO	**0.9922**	**0.9963**	**0.9871**	0.0023	0.6071	0.79029	0.8499	0.0182
	PROWSYN	0.9667	0.9839	0.9837	0.0031	**0.6198**	**0.7931**	**0.8557**	0.0389
Vehicle	ADASYN	0.8844	0.9265	0.9628	0.0187	0.9072	0.9603	0.9764	0.012
	RAMO	**0.8884**	0.9277	0.9591	0.0228	0.9108	**0.9617**	**0.9801**	0.0062
	CBSO	0.8761	0.9251	0.961	0.017	0.8818	0.9496	0.9673	0.0205
	PROWSYN	0.8832	**0.9283**	**0.9634**	0.02	**0.9120**	0.9556	0.9726	0.0147
Btissue	ADASYN	0.6671	0.7286	0.8103	0.1408	0.6557	0.7055	0.7966	0.1369
	RAMO	0.7149	0.7754	0.8754	0.0903	0.639	0.6897	0.809	0.1081
	CBSO	0.5857	0.6693	0.8004	0.1327	0.6533	0.6732	0.7933	0.1027
	PROWSYN	**0.7805**	**0.8328**	**0.8985**	0.0876	**0.6844**	**0.7213**	**0.8482**	0.0759
Pima	ADASYN	0.5536	0.6471	0.7382	0.0452	0.6643	0.7181	**0.8165**	0.0459
	RAMO	0.5772	0.6669	0.75	0.0227	0.65	0.7034	0.7894	0.0762
	CBSO	0.5836	0.6707	0.7427	0.0437	0.656	0.7092	0.8144	0.04
	PROWSYN	**0.5962**	**0.6810**	**0.7611**	0.0541	**0.6883**	**0.7502**	0.8125	0.0361
Win Time	ADASYN	0	1	0		0	1	1	
	RAMO	2	1	1		1	2	2	
	CBSO	3	2	2		1	2	1	
	PROWSYN	5	6	7		8	5	6	

Table 3. Detailed computation of Wilcoxon test [24] statistic on AUC results of PROWSYN vs. ADASYN [17] for Decision Tree Classifier

Dataset	PROWSYN	ADASYN	Difference	Rank
PageBlocks	0.98573	0.98338	0.00235	+2
Abalone	0.75282	0.74956	0.00326	+4
CG	0.91376	0.89108	0.02268	+7
Segment	0.9665	0.96093	0.00557	+5
Libra	0.90546	0.85464	0.05082	+9
Yeast	0.88657	0.87287	0.0137	+6
Robot	0.98374	0.98636	-0.00262	-3
Vehicle	0.96346	0.96288	0.00058	+1
Btissue	0.89857	0.81033	0.08824	+10
Pima	0.76118	0.73829	0.02289	+8
	$R+ = 52$, $R- = 3$, $T = min\{52, 3\} = 3$			

Table 4. Significance tests of averaged AUC between PROWSYN vs. ADASYN [17], RAMO [18], and CBSO [19] for Decision Tree and Neural Network Classifiers. All results are significant (less than or equal to critical value 8) except ProWSyn vs. RAMO for Neural Network classifier ($T = 10$ is larger than critical value 8)

	Decision Tree Classifier			Neural Network Classifier		
	PROWSYN vs.			PROWSYN vs.		
	ADASYN	RAMO	CBSO	ADASYN	RAMO	CBSO
$R+$	52	48	47	52	45	50
$R-$	3	7	8	3	10	5
T	3	7	8	3	10	5

ProwSyn with each of the other methods in a pairwise manner. For 10 datasets, the test statistic, i.e. T should be less than or equal to critical value 8 [25] to reject the null hypothesis at the significance level of 0.05. Table 3 shows the detailed computation of the Wilcoxon statistic, i.e. T for ProsSyn vs. ADASYN results of Decision tree classifier. The obtained value of $T = 3$ is less than the critical value 8. This proves that ProwSyn is statistically better than the ADASYN. For space consideration, we avoid the detailed computation and show only the test statistic values for all other comparisons in Table 4. We see that ProwSyn statistically outperforms other methods for all comparisons except ProWSyn vs. RAMO for neural network classifier ($T = 10$ is larger than critical value 8). In this case, ProWSyn can not statistically outperform RAMO. However, in Table 2, AUC column under Neural Network Classifier shows that ProWSyn wins in 6 cases while RAMO wins 2 cases. This difference in winning time shows that ProWSyn performs better than RAMO for neural network classifier.

5 Conclusion

In this paper, we try to identify the problems related to the synthetic sample generation process of oversampling methods in dealing with imbalanced data sets. Existing algorithms [17–19] generate inaccurate weights for the minority samples that results in very poor and skewed distribution of synthetic samples (Sect. 2). We thus present a new synthetic oversampling algorithm, ProWSyn, for balancing the majority and minority class distributions in an imbalanced data set. Our ProWSyn avoids the aforementioned problem by assigning effective weights to the minority class samples using their Euclidean distance from the majority class samples in the data set. ProWSyn partitions the minority data set into several partitions based on samples' proximity from the decision boundary. and uses this proximity information for the minority samples in such a way that the closest sample get highest weight and the furthest ones get the lowest weight. By doing so, our method ensures a proper distribution of weights among the minority samples according to their position from the decision boundary. This results in a effective distribution of generated synthetic samples across the minority data set. The simulation results show that ProWSyn can statistically outperform ADASYN, CBSO, and RAMO algorithms in terms of a number of performance metrics such as AUC, F-measure, and G-mean. Several other research issues can be investigated using ProWSyn such as application of ProWSyn in multi-class problems, integration of ProWSyn with some other undersampling methods, and integration of ProWSyn with an ensemble technique such as Adaboost.M2 boosting ensemble.

Acknowledgments. The work has been done in the Computer Science & Engineering Department of Bangladesh University of Engineering and Technology (BUET). The authors would like to acknowledge BUET for its generous support.

References

1. Weiss, G.M.: Mining with Rarity: A Unifying Framework. ACM SIGKDD Explorations Newsletter 6(1), 7–19 (2004)
2. Holte, R.C., Acker, L., Porter, B.W.: Concept Learning and the Problem of Small Disjuncts. In: Proc. Int'l J. Conf. Artificial Intelligence, pp. 813–818 (1989)
3. Quinlan, J.R.: Induction of Decision Trees. Machine Learning 1(1), 81–106 (1986)
4. Murphy, P.M., Aha, D.W.: UCI repository of Machine learning databases. University of California Irvine, Department of Information and Computer Science
5. Lewis, D., Catlett, J.: Heterogeneous Uncertainty Sampling for Supervised Learning. In: Proc. of the Eleventh International Conference of Machine Learning, pp. 148–156 (1994)
6. Fawcett, T.E., Provost, F.: Adaptive Fraud Detection. Data Mining and Knowledge Discovery 3(1), 291–316 (1997)
7. Kubat, M., Holte, R.C., Matwin, S.: Machine Learning for the Detection of Oil Spills in Satellite Radar Images. Machine Learning 30(2/3), 195–215 (1998)
8. Ling, C.X., Li, C.: Data Mining for Direct Marketing: Problems and Solutions. In: Proc. Int'l Conf. on Knowledge Discovery & Data Mining (1998)

9. Japkowicz, N., Myers, C., Gluck, M.: A Novelty Detection Approach to Classification. In: Proc. of the Fourteenth International Joint Conference on Artificial Intelligence, pp. 518–523 (1995)
10. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. 21(10), 1263–1284 (2009)
11. Liu, X.Y., Wu, J., Zhou, Z.H.: Exploratory Under Sampling for Class Imbalance Learning. In: Proc. Int'l Conf. Data Mining, pp. 965–969 (2006)
12. Zhang, J., Mani, I.: KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. In: Proc. Int'l Conf. Machine Learning, ICML 2003, Workshop Learning from Imbalanced Data Sets (2003)
13. Kubat, M., Matwin, S.: Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In: Proc. Int'l Conf. Machine Learning, pp. 179–186 (1997)
14. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. ACM SIGKDD Explorations Newsletter 6(1), 20–29 (2004)
15. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic Minority Over-Sampling Technique. J. Artificial Intelligence Research 16, 321–357 (2002)
16. Cieslak, D.A., Chawla, N.V.: Start Globally, Optimize Locally, Predict Globally: Improving Performance on Imbalanced Data. In: Proc. IEEE Int'l Conf. Data Mining, pp. 143–152 (2008)
17. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. In: Proc. Int'l J. Conf. Neural Networks, pp. 1322–1328 (2008)
18. Chen, S., He, H., Garcia, E.A.: RAMOBoost: Ranked Minority Oversampling in Boosting. IEEE Trans. Neural Networks 21(20), 1624–1642 (2010)
19. Barua, S., Islam, M. M., Murase, K.: A Novel Synthetic Minority Oversampling Technique for Imbalanced Data Set Learning. In: Lu, B.-L., Zhang, L., Kwok, J. (eds.) ICONIP 2011, Part II. LNCS, vol. 7063, pp. 735–744. Springer, Heidelberg (2011)
20. Japkowicz, N., Stephen, S.: The Class Imbalance Problem: A Systematic Study. Intelligent Data Analysis 6(5), 429–449 (2000)
21. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann, San Francisco (1993)
22. UCI Machine Learning Repository, `http://archive.ics.uci.edu/ml/`
23. Fawcett, T.: ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. Technical Report HPL-2003-4, HP Labs (2003)
24. Corder, G.W., Foreman, D.I.: Nonparametric Statistics for Non-Statisticians: A step-by-Step Approach. Wiley, New York (2009)
25. Critical Value Table of Wilcoxon Signed-Ranks Test, `http://www.sussex.ac.uk/Users/grahamh/RM1web/WilcoxonTable2005.pdf`