Applying Support Vector Machines to Imbalanced Datasets

Rehan Akbani[1], Stephen Kwek[1], and Nathalie Japkowicz[2]

[1] Department of Computer Science, University of Texas at San Antonio
6900 N. Loop 1604 W, San Antonio, Texas, 78249, USA
{rakbani,kwek}@cs.utsa.edu
[2] School of Information Technology & Engineering, University of Ottawa
150 Louis Pasteur, Ottawa, Ontario, K1N 6N5, Canada
nat@site.uottawa.ca

Abstract. Support Vector Machines (SVM) have been extensively studied and have shown remarkable success in many applications. However the success of SVM is very limited when it is applied to the problem of learning from imbalanced datasets in which negative instances heavily outnumber the positive instances (e.g. in gene profiling and detecting credit card fraud). This paper discusses the factors behind this failure and explains why the common strategy of undersampling the training data may not be the best choice for SVM. We then propose an algorithm for overcoming these problems which is based on a variant of the SMOTE algorithm by Chawla et al, combined with Veropoulos et al's different error costs algorithm. We compare the performance of our algorithm against these two algorithms, along with undersampling and regular SVM and show that our algorithm outperforms all of them.

1 Introduction

Support Vector Machines (SVM) were introduced by Vapnik and colleagues [13] and they have been very successful in application areas ranging from image retrieval [12], handwriting recognition [3] to text classification [7]. However, when faced with imbalanced datasets where the number of negative instances far outnumbers the positive instances, the performance of SVM drops significantly [15] (in the remainder of this paper negative is always taken to be the majority class and positive is the minority class). Application areas such as gene profiling, medical diagnosis and credit card fraud detection have highly skewed datasets with a very small number of positive instances which are hard to classify correctly, but important to detect nevertheless [15]. An imbalance of 100 to 1 exists in fraud detection domains, even approaching 100,000 to 1 in other applications [11].

Classifiers generally perform poorly on imbalanced datasets because they are designed to generalize from sample data and output the simplest hypothesis that best fits the data, based on the principle of Occam's razor. This principle is embedded in the inductive bias of many machine learning algorithms including decision trees, which favor shorter trees over longer ones. With imbalanced data, the simplest hypothesis is often the one that classifies almost all instances as negative.

J.-F. Boulicaut et al. (Eds.): ECML 2004, LNAI 3201, pp. 39–50, 2004.

Another factor is that making the classifier too specific may make it too sensitive to noise and more prone to learn an erroneous hypothesis. Certain algorithms specifically modify the behavior of existing algorithms to make them more immune to noisy instances, such as the IB3 algorithm [1] for kNN, or pruning of decision trees, or soft margins in SVM [13]. While these approaches work well for balanced datasets, with highly imbalanced datasets having ratios of 50 to 1 or more the simplest hypothesis is often the one that classifies every instance as negative. Furthermore, the positive instances can be treated as noise and ignored completely by the classifier.

A popular approach towards solving these problems is to bias the classifier so that it pays more attention to the positive instances. This can be done, for instance, by increasing the penalty associated with misclassifying the positive class relative to the negative class. Another approach is to preprocess the data by oversampling the majority class or undersampling the minority class in order to create a balanced dataset.

We combine both of these approaches in our algorithm and show that we can significantly improve the performance of SVM compared to applying any one approach. We also show in this paper that even though undersampling the majority class does improve SVM performance, there is an inherent loss of valuable information in this process. Our goal was to retain and use this valuable information, while simultaneously boosting the efficacy of oversampled data. Combined with this dual approach we also used a bias to make SVM more sensitive to the positive class.

We specifically chose SVM to attack the problem of imbalanced data because SVM is based on strong theoretical foundations [13] and our empirical results show that it performs well with moderately imbalanced data even without any modifications. Its unique learning mechanism makes it an interesting candidate for dealing with imbalanced datasets, since SVM only takes into account those instances that are close to the boundary, i.e. the support vectors, for building its model. This means that SVM is unaffected by non-noisy negative instances far away from the boundary even if they are huge in number.

In Section 2 we outline related work dealing with the problem of imbalanced data. Section 3 investigates the effects of imbalance on SVM, while Section 4 discusses the problems associated with undersampling the majority class. Section 5 presents our approach to the problem and describes our technique (SDC) of combining SMOTE [2] with Different Costs [14]. Finally Section 6 gives the conclusions.

2 Related Work

The problem of imbalanced datasets has been approached from two main directions. The first approach is to preprocess the data by undersampling the majority instances or oversampling the minority instances. Kubat and Matwin [9] proposed a one-sided selection process which undersampled the majority class in order to remove noisy, borderline, and redundant training instances. But for the specific case of SVM, removing redundant (far away) instances has no effect and removing borderline instances may adversely affect the accuracy of the learned hyperplane.

Japkowicz [6] evaluated the oversampling and undersampling techniques for skewed datasets and concluded that both methods were effective. Ling and Li [10] combined oversampling with undersampling, but this combination did not provide significant improvement in the "lift index" metric that they used.

Chawla et al [2] devised a method called Synthetic Minority Oversampling Technique (SMOTE). This technique involved creating new instances through "phantom-transduction." For each positive instance, its nearest positive neighbors were identified and new positive instances were created and placed randomly in between the instance and its neighbors. Since this technique creates new positive instances, we found this technique to be more useful for SVM than simple oversampling.

The other approach to dealing with imbalanced datasets using SVM biases the algorithm so that the learned hyperplane is further away from the positive class. This is done in order to compensate for the skew associated with imbalanced datasets which pushes the hyperplane closer to the positive class. This biasing can be accomplished in various ways. In [15] an algorithm is proposed that changes the kernel function to develop this bias, while in [4] the kernel matrix is adjusted to fit the training data. Veropoulos et al [14] suggested using different penalty constants for different classes of data, making errors on positive instances costlier than errors on negative instances. In this paper we will combine this method together with SMOTE [2] to develop a classifier that performs better than either of these algorithms alone.

3 Effects of Imbalance on SVM

Given a set of labeled instances $X_{train} = \{x_i, \ y_i\}^n{}_{i=1}$ and a kernel function K, SVM finds the optimal α_i for each x_i to maximize the margin γ between the hyperplane and the closest instances to it. The class prediction for a new test instance x is made through:

$$sign\left(f(x) = \sum_{i=1}^{n} y_i \alpha_i K(x, x_i) + b \right) \tag{1}$$

where b is the threshold. 1-norm soft-margin SVMs minimize the primal Lagrangian:

$$L_p = \frac{\|w\|^2}{2} + C\sum_{i=1}^{n} \xi_i \ - \sum_{i=1}^{n} \alpha_i \left[y_i \left(w \ . \ x_i + b \right) - 1 + \xi_i \right] - \sum_{i=1}^{n} r_i \xi_i \tag{2}$$

where $\alpha_i \geq 0$ and $r_i \geq 0$ [5]. The penalty constant C represents the trade-off between the empirical error ξ and the margin. In order to meet the Karush-Kuhn-Tucker (KKT) conditions, the value of α_i must satisfy:

$$0 \leq \alpha_i \leq C \ \text{ and } \ \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{3}$$

3.1 Causes of Performance Loss with Imbalanced Data

1. Positive Points Lie Further from the Ideal Boundary. Wu and Chang [15] point out this factor as one source of boundary skew. They mention that the imbalance in the training data ratio means that the positive instances may lie further away from the "ideal" boundary than the negative instances. This is illustrated by way of example that if we were to draw n randomly chosen numbers between 1 to 100 from a uniform distribution, our chances of drawing a number close to 100 would improve with increasing values of n, even though the expected mean of the draws is invariant of n. As a result of this phenomenon, SVM learns a boundary that is too close to and skewed towards the positive instances.

2. Weakness of Soft-Margins. Mathematically, we can see from eq. 2 that minimizing the first term on the right hand side $\|w\|^2/2$, is equivalent to maximizing the margin γ, while minimizing the second term $C\sum\xi$ minimizes the associated error. The constant C specifies what tradeoff we are willing to tolerate between maximizing the margin and minimizing the error. If C is not very large, SVM simply learns to classify everything as negative because that makes the "margin" the largest, with zero cumulative error on the abundant negative examples. The only tradeoff is the small amount of cumulative error on the few positive examples, which does not count for much. This explains why SVM fails completely in situations with a high degree of imbalance. One way to combat this is to increase the tradeoff C^+ associated with the positive examples. This is exactly what Veropoulos et al [14] propose in their paper and this is the strategy we adopt in this paper as well (more about this in Section 5).

3. Imbalanced Support Vector Ratio. Another source of boundary skew according to Wu and Chang [15] is the imbalanced support vector ratio. They found that as the training data gets more imbalanced, the ratio between the positive and negative support vectors also becomes more imbalanced. They hypothesize that as a result of this imbalance, the neighborhood of a test instance close to the boundary is more likely to be dominated by negative support vectors and hence the decision function is more likely to classify a boundary point negative. We would like to point out however, that because of the KKT conditions in eq. 3, the sum of the α's associated with the positive support vectors must be equal to the sum of the α's associated with the negative support vectors. Because there are fewer positive support vectors with correspondingly fewer α's, each positive support vector's α must be larger than the negative support vector's α on average. These α's act as weights in the final decision function (eq. 1) and as a result of larger α's the positive support vectors receive a higher weight than the negative support vectors which offsets the effect of support vector imbalance to some extent. This shows why SVM does not perform too badly compared to other machine learning algorithms for moderately skewed datasets.

4 Problems with Undersampling

Undersampling of the majority class is a popular method for dealing with imbalanced datasets. The rationale behind it is to try to balance out the dataset in an attempt to overcome the idiosyncrasies of the machine learning algorithm. The problem with this approach, however, is that the purpose of machine learning is for the classifier to estimate the probability distribution of the target population. Since that distribution is unknown we try to estimate the population distribution using a sample distribution. Statistics tells us that as long as the sample is drawn randomly, the sample distribution can be used to estimate the population distribution from where it was drawn. Hence, by learning the sample distribution we can learn to approximate the target distribution. Once we perform undersampling of the majority class, however, the sample can no longer be considered random.

A possible defense against this argument is when we assume that in an imbalanced dataset the sample was not drawn randomly to begin with. The assumption is that the sampling mechanism was unfairly biased towards sampling the majority instances. For instance, in detecting credit card fraud, the default assumption is that a transaction is valid unless proven otherwise. As a result, many fraudulent transactions may go undetected by the source labeling the dataset. Similarly, in medical diagnosis, a person with a rare disease may not be properly diagnosed. To counter these inevitable deficiencies, undersampling or oversampling is done to overcome the biases of the sampling mechanism. Even though it is impossible for undersampling or oversampling to make a non-random sample random, in practice these measures have empirically been shown to approximate the target population better than the original, biased sample.

The second problem with undersampling is that we are throwing away valid instances, which contain valuable information. The nature of the information these instances contain can be understood in the following way. The problem with imbalanced datasets is that they skew the boundary towards the positive instances (Figure 1). The classification function for the hard-margin linear SVM [5] is:

$$sign\left(\langle w \ . \ x \rangle + b \right) \tag{4}$$

Where *w* is a vector that is normal to the separating hyperplane. The norm of *w* and the variable *b* decide the distance of the hyperplane from the origin. With non-linear SVMs, the kernel function maps the data into a high-dimensional *feature space* where a hyperplane is used to separate the data. Any hyperplane can be defined by its orientation, given by the direction of *w*, and its distance from the origin. The task of SVM is to learn the optimal hyperplane in the feature space. In order to do this, it takes cues from the dataset about the orientation and distance of the optimal hyperplane.

We hypothesized that, given a relatively noise-free but imbalanced dataset that is linearly separable in the feature space, SVM will learn to approximate the orientation of the hyperplane better than using the same dataset after it is undersampled. Consider Figure 1. We conducted some experiments with artificial data in which we first defined a boundary and called it the "ideal boundary." We then generated several instances at random above the boundary and labeled them as negative. We also randomly generated a few instances below the boundary and labeled them as positive. The number of negative instances generated far outnumbered the positive instances to simulate imbalanced data. Note that in Figure 1, the negative instances lie much closer to the ideal boundary than the positive instances due to reasons given in Section 2. If, given the dataset, SVM learned the ideal boundary then it would be able to classify all the instances and any test instances perfectly. But as expected, SVM learned a boundary that was midway between the positive and negative support vectors and therefore, much further from the ideal boundary. Such a boundary would not be able to classify test instances very well. Figure 1 and Figure 2 represent some typical results we obtained with our experiments on artificial datasets.

In Figure 1, SVM can obtain reasonable cues about the orientation of the hyperplane from the negative instances that lie close to the boundary. As a result, the learned hyperplane has almost the same orientation as the ideal hyperplane. The only problem is that the distance of the learned hyperplane is too far off to the positive side. In Figure 2, however, the majority instances have been randomly undersampled until their numbers are about equal to the minority instances. In this case, we can see that the negative and positive instances are approximately the same distance away from the ideal hyperplane, making the distance of the learned hyperplane very close to the distance of the ideal hyperplane. But now the problem is that the negative instances can no longer give good cues about the orientation of the hyperplane and there is a greater degree of freedom for the orientation of the hyperplane to vary.

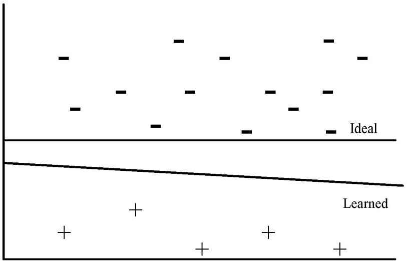

Fig. 1. Positive instances lie further away from the ideal boundary (*horizontal line*) than the negative instances. As a result SVM learns a boundary (*slanted line*) that is too close to the positive support vectors

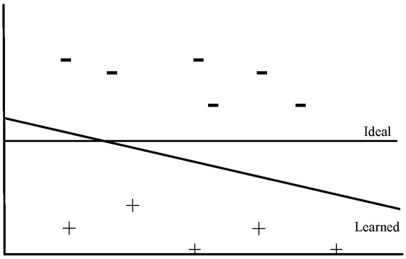

Fig. 2. After undersampling the minority instances, the learned plane estimates the distance of the ideal plane better than in Figure 1, but the orientation of the learned plane is no longer as accurate

We decided to test this hypothesis on UCI datasets. In order to know what the ideal boundary for these datasets was, it was necessary to start with large, balanced datasets that were relatively noise-free and linearly separable in the feature space. We chose five balanced UCI datasets for this experiment *balance3* (RBF), *chess1*(RBF), *mushroom1* (linear), *ionosphere1* (RBF) and *sonar1* (polynomial). The number in the suffix represents the class we chose to be the positive class while the kernel function is indicated in parenthesis. We trained SVM using three different kernels on these datasets (linear, polynomial with degree 2 and Radial Basis Function with gamma =1), and in each case we obtained nearly 100% accuracy (which is why we chose these datasets). The hyperplane learned was assumed to be the ideal hyperplane i.e. if SVM were to learn this hyperplane then it would be able to classify all instances almost perfectly. We then randomly removed most of the positive instances from each dataset to make them highly imbalanced. We re-trained SVM using these imbalanced datasets to obtain the learned boundary. Next we obtained the dot product of the two vectors *w* (obtained from the balanced dataset model) and *v* (the vector obtained from the imbalanced dataset model) and used this dot product to compute *θ*, the angle between the ideal and the learned hyperplanes. We obtained *w* from the learned model by applying the following equation for linear SVMs [5]:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{5}$$

Each of the three parameters *α, y* and *x* (the support vectors) can be obtained directly from the learned model. When using non-linear kernels, calculating *w* in this way may not represent the absolute angle of the hyperplane in the feature space. However, we are only interested in knowing how different the shape of the boundary is in the balanced vs. the imbalanced case. Computing *w* in this way and then computing the angle between *w* and *v* allows us to estimate this difference in shape.

We repeated this experiment by keeping the positive instances the same as in the imbalanced datasets and reducing the negative instances by gradually randomly undersampling them until eventually the negative instances were equal in number to the positive instances. In each case we measured the angle between the ideal hyperplane and the learned hyperplane. The results are shown below.

Table 1. Angle (in degrees) between the ideal hyperplane and the learned hyperplane as the imbalance ratio (*ratio of negative to positive examples*) is varied

Imbalance	Balance	Chess	Mushroom	Ionosphere	Sonar
15	10.4	24.88	58.1	35.06	65.65
8	10.88	24.87	58.6	36.08	66.32
4	10.57	25.25	58.9	46.77	70.48
2	17.44	28.2	64.58	55.97	79.13
1	26.2	31.92	66.1	65.6	82.76

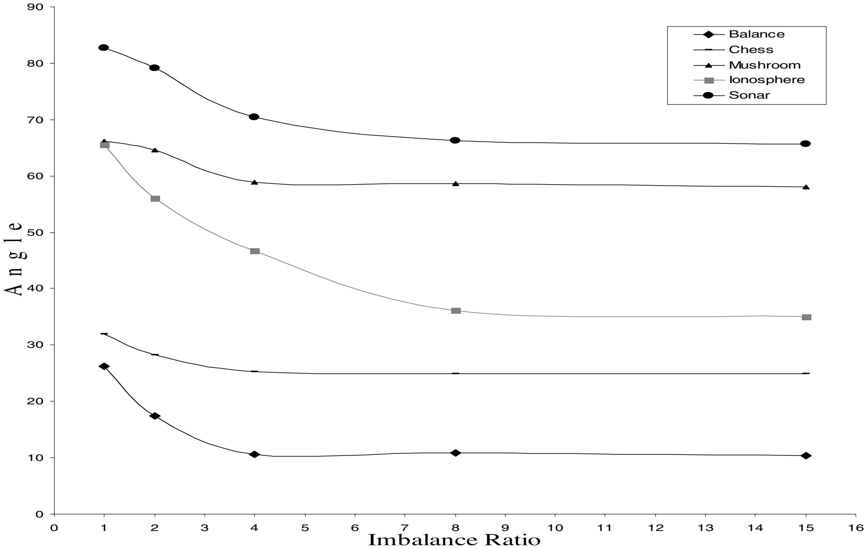

Fig. 3. Graph showing the effect of varying the imbalance ratio (*x*-axis) on angle between the ideal and the learned hyperplane (*y-axis*). The angle is smaller in more imbalanced datasets

The results agree with our hypothesis that undersampling the majority class causes larger angles between the ideal and learned hyperplane due to the reasons given earlier in this section. Undersampling also reduces the total number of training instances which also contributes to increasing angles. Therefore, any benefit from undersampling occurs mainly because of a more accurate estimate of the distance, rather than the orientation, of the hyperplane from the ideal boundary. Oversampling of the minority instances, on the other hand, does not lead to loss of information. Even though it may not make the non-random sample truly random (nothing really can), it has been shown, nevertheless, to make the classifier perform better than with the regular imbalanced sample [6]. As a result we will employ oversampling in our approach.

5 Our Approach – SMOTE with Different Costs (SDC)

As mentioned above, undersampling data has its drawbacks and results in information loss. We would like to devise an approach that keeps the majority instances and yet performs well with imbalanced data. The problem is that with imbalanced datasets, the learned boundary is too close to the positive instances. We need to bias SVM in a way that will push the boundary away from the positive instances. Veropoulos et al [14] suggest using different error costs for the positive (C^+) and negative (C^-) classes. Specifically, they suggest changing the primal Lagrangian (eq. 2) to:

$$L_p = \frac{\|w\|^2}{2} + C^+ \sum_{\{i|yi=+1\}}^{n_+} \xi_i + C^- \sum_{\{j|yj=-1\}}^{n_-} \xi_j - \sum_{i=1}^{n} \alpha_i \left[y_i \left(w \; . \; x_i + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{n} r_i \xi_i \tag{6}$$

The constraints on α_i then become:

$$0 \le \alpha_i \le C^+ \text{ if } y_i = +1 \quad \text{and} \quad 0 \le \alpha_i \le C^- \text{ if } y_i = -1 \tag{7}$$

Furthermore, we note that $\xi_i > 0$ only when $\alpha_i = C$ [5]. Therefore non-zero errors on positive support vectors will have larger α_i while non-zero errors on negative support vectors will have smaller α_i. The net effect is that the boundary is pushed more towards the negative instances. However, a consequence of this is that SVM becomes more sensitive to the positive instances and obtains stronger cues from the positive instances about the orientation of the plane than from the negative instances. If the positive instances are sparse, as in imbalanced datasets, then the boundary may not have the proper shape in the input space as illustrated in Figure 4.

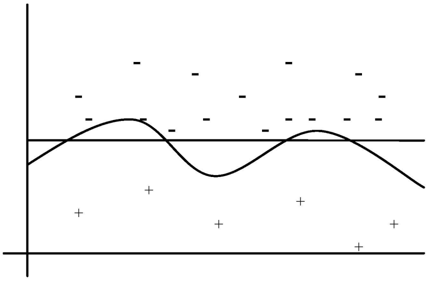

Fig. 4. The learned boundary (*curved line*) in the input space closely follows the distribution of the positive instances. The ideal boundary is denoted by the horizontal line

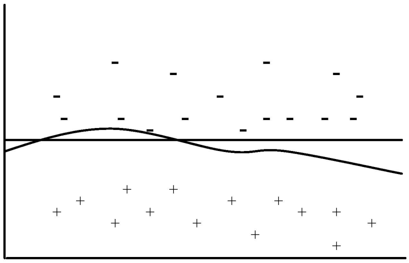

Fig. 5. After using SMOTE, the positive instances are now more densely distributed and the learned boundary (*curved line*) is more well defined

The solution we adopted to remedy the problem of sparse positive instances is to use Chawla et al's [2] SMOTE algorithm mentioned in Section 2 of this paper. Using the SMOTE technique of oversampling the minority instances, we can make the distribution of positive instances denser. Simply resampling the minority instances merely overlaps the positive instances on top of each other and does not help in "smoothing out" the shape of the boundary. SMOTE synthetically generates new instances between two existing positive instances which helps in making their distribution more well-defined. After using SMOTE, the input space may look like Figure 5.

Therefore, in summary, our strategy consists of:

1. Not undersampling the majority instances since they lead to loss of information.
2. Using different error costs for different classes to push the boundary away from the positive instances.
3. Using SMOTE to make the positive instances more densely distributed in order to make the boundary more well defined.

5.1 Experiments and Results

In order to evaluate classifiers on highly imbalanced datasets, using accuracy as a metric is virtually useless. This is because with an imbalance of 99 to 1, a classifier that classifies everything negative will be 99% accurate, but it will be completely useless as a classifier. The medical community, and increasingly the machine learning community [14, 15], use two metrics, the sensitivity and the specificity, when evaluating the performance of various tests. Sensitivity can be defined as the accuracy on the positive instances (true positives / (true positives + false negatives)), while specificity can be defined as the accuracy on the negative instances (true negatives / (true negatives + false positives)). Kubat et al [9] suggested the g-means metric defined as:

$$g = \sqrt{acc+ \ . \ acc-} \tag{8}$$

Where acc+ = sensitivity and acc- = specificity. This metric has been used by several researchers for evaluating classifiers on imbalanced datasets [8, 9, 15]. We will also use this metric to evaluate our classifier. We also list the sensitivity and specificity separately to give the reader an even better idea of the performance of our classifier.

In our experiments, we compared the performance of our classifier with regular SVM, random undersampling [6], SMOTE [2], and different error costs [14]. We used 10 different UCI datasets with varying degrees of class imbalance (Table 2). Each dataset was randomly split into train and test sets in the ratio 7 to 3, while sampling them in a stratified manner to ensure each of them had the same negative to positive ratio [9, 15]. For the undersampling algorithm, we undersampled the training data until both the classes were equal in number as Japkowicz [6] did in her experiments. For SMOTE [2] we oversampled the data instead of undersampling it. The amount of oversampling is given in Table 2 below. Finally, for the different error costs algorithm, Veropoulos et al [14] have not suggested any guidelines for deciding what the relative ratios of the positive to negative cost factors should be. As a rule of thumb, we empirically found that setting the cost ratio to the inverse of the imbalance ratio gave good results and that is what we have used. The results of our experiments are given below.

Table 2. The table below lists the UCI datasets we used, the kernel function used (based on the best empirical results), the number of positive instances and the number of negative instances in the dataset. It also lists the amount of oversampling of the minority class we did for SMOTE and in our algorithm, SMOTE with Different Costs (SDC), based on the amount of imbalance present in the original dataset. The suffix after each dataset indicates the class we used as the positive class. RBF is the radial basis function with gamma = 1, while the polynomial kernel has degree = 2

Dataset	Kernel	Positive Insts.	Negative Insts.	% Oversampled
Abalone19	Linear	32	4145	1000
Anneal5	Linear	67	831	400
Car3	RBF	69	1659	400
Glass7	RBF	29	185	200
Hepatitis1	Linear	32	123	100
Hypothyroid3	Linear	95	3677	400
Letter26	RBF	734	19266	200
Segment1	RBF	330	1980	100
Sick2	Polynomial	231	3541	100
Soybean12	Linear	44	639	100

Table 3. The table below shows the sensitivity (Se) and Specificity (Sp) of each algorithm. US stands for UnderSampling, DEC stands for Different Error Costs [14], while our algorithm is SDC (SMOTE with Different Costs)

	SVM		US		SMOTE		DEC		SDC	
	Se	Sp	Se	Sp	Se	Sp	Se	Sp	Se	Sp
abalone	0	1	0.778	0.533	0	1	0.889	0.732	0.808	0.687
anneal	1	1	1	1	1	1	1	1	1	1
car	0	1	0.95	0.97	0.97	1	0.1	1	0.97	1
glass	0.75	1	0.875	0.885	0.769	1	0.875	0.967	0.808	1
hepatitis	0.364	0.977	0.727	0.767	0.625	0.881	0.545	0.884	0.708	0.833
hypothy	0.031	1	0.906	0.882	0.646	0.997	0.906	0.972	0.957	0.96
letter	0.67	1	0.996	0.917	0.903	0.995	1	0.975	0.997	0.966
segment	0.99	1	0.99	0.993	0.954	1	0.99	1	0.959	0.998
sick	0	1	0.773	0.756	0.167	0.995	0.864	0.862	0.865	0.874
soybean	0.857	1	1	0.936	1	1	1	1	1	1

Note that SVM has almost perfect specificity, but poor sensitivity because it tends to classify everything as negative. Any algorithm that tries to improve on it inevitably sacrifices some specificity in order to improve the sensitivity. That is why, Kubat and Matwin's [9] g-means metric (Table 4) is the best of the three measures because it combines both the sensitivity and the specificity and takes their geometric mean.

Table 4. The table below shows the performance of the five algorithms using Kubat and Matwin's[9] g-means metric. The last line of the table is the arithmetic mean of each algorithm over all the g-means metrics. This arithmetic mean can be used to quantify the overall performance of each algorithm over all ten datasets

	SVM	US	SMOTE	DEC	SDC
abalone	0	0.6436394	0	0.8064562	0.7449049
anneal	1	1	1	1	1
car	0	0.960104	0.9846381	0.3162278	0.9846381
glass	0.8660254	0.880108	0.877058	0.9199519	0.9405399
hepatitis	0.5959695	0.7470874	0.742021	0.6942835	0.7682954
hypothyroi	0.1767767	0.8938961	0.8025625	0.9384492	0.9581446
letter	0.8182931	0.9555176	0.947737	0.9871834	0.9816909
segment	0.9950372	0.9917748	0.9765287	0.9950372	0.9783467
sick	0	0.7641141	0.4071283	0.8627879	0.8695146
soybean	0.9258201	0.9672867	1	1	1
Mean	**0.537792**	**0.880353**	**0.773767**	**0.852038**	**0.922608**

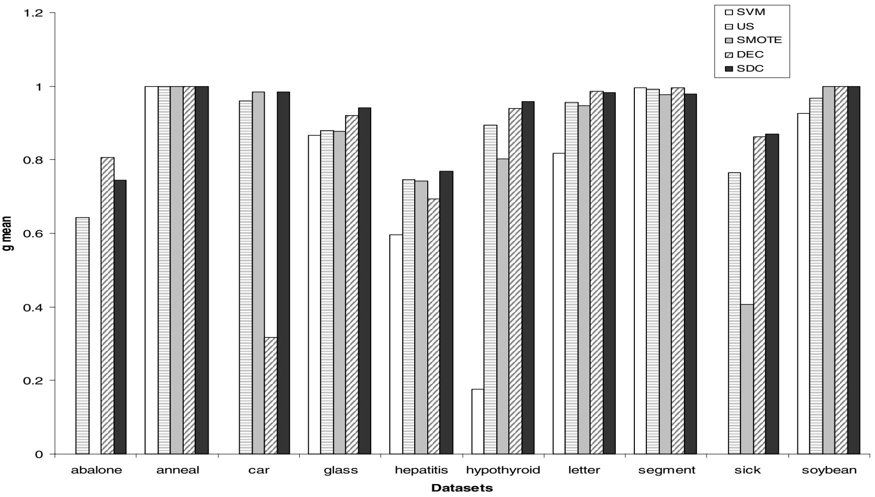

Fig. 6. g-means graphs for the datasets shown in Table 4. The five algorithms, in order, are SVM, Undersampling, SMOTE, Different Error Costs and the last bar is our algorithm SDC

6 Conclusion

The results show that our SDC algorithm outperforms all the other four algorithms. In seven out of the ten datasets our algorithm has the highest g-means metric, and in the remaining three datasets it is not lower by much. It should be noted that our approach never performs worse than SMOTE. In the three cases where our approach performs worse than DEC or other algorithms, it is probably due to the fact that SMOTE itself makes some assumptions about the training set. For instance, it assumes that the space between two positive instances is assumed to be positive and the neighborhood of a positive instance is also assumed to be positive [15], which may not always be true.

Since our algorithm uses SMOTE, it also makes a similar assumption. In datasets where this assumption may not hold, our algorithm will perform slightly worse than the other algorithms.

Undersampling does show significant performance gain, but as noted before the gain is mainly due to accurate estimation of the hyperplane's distance, not its orientation. As a result, it has the highest score in only one dataset where every other algorithm also has a perfect score (anneal). Our algorithm, on the other hand, tries to estimate not only the correct distance but also the correct orientation of the hyperplane and thus performs better than the rest, even in cases where SVM fails completely.

References

1. Aha, D. (1992). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. International Journal Man-Machine Studies, 36, 267-287.
2. Chawla, N., Bowyer, K., Hall, L. & Kegelmeyer, W. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321-357.
3. Cortes, C. (1995). Prediction of Generalisation Ability in Learning Machines. PhD thesis, Department of Computer Science, University of Rochester.
4. Cristianini, N., Kandola, J., Elisseeff, A. & Shawe-Taylor, J. (2002). On kernel target alignment. Journal Machine Learning Research, 1.
5. Cristianini, N. & Shawe-Taylor, J. (2000). An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, Cambridge, UK.
6. 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, Nevada.
7. Joachims, T. (1998). Text Categorization with SVM: Learning with Many Relevant Features. Proceedings of ECML-98, 10th European Conference on Machine Learning.
8. Kubat, M., Holte, R. & Matwin, S. (1997). Learning when Negative Examples Abound. In Proceedings of ECML-97, 9th European Conference on Machine Learning.
9. Kubat, M. & Matwin, S. (1997). Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. Proceedings of the 14th International Conference on Machine Learning.
10. 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, New York.
11. Provost, F. & Fawcett, T. (2001). Robust Classification for Imprecise Environments. Machine Learning, 42/3, 203-231.
12. Tong, S. & Chang, E. (2001). Support Vector Machine Active Learning for Image Retrieval. Proceedings of ACM International Conference on Multimedia, 107-118.
13. Vapnik, V. (1995). The Nature of Statistical Learning Theory. Springer-Verlag, NY.
14. Veropoulos, K., Campbell, C., & Cristianini, N. (1999). Controlling the sensitivity of support vector machines. Proceedings of the International Joint Conference on AI, 55–60.
15. 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.