Improving Classification with Support Vector Machines

Maria Muntean*, Honoriu Valean, Ioan Ileana*, Corina Rotar***

**1 Decembrie 1918 University of Alba Iulia, Romania*
(e-mail: mmuntean@ uab.ro, iileana@uab.ro, crotar@uab.ro)
***2Technical University of Cluj Napoca, Romania*
(e-mail: Honoriu.Valean@aut.utcluj.ro)

Abstract: A problem arises in data mining, when classifying unbalanced datasets using Support Vector Machines. Because of the uneven distribution and the soft margin of the classifier, the algorithm tries to improve the general accuracy of classifying a dataset, and in this process it might misclassify a lot of weakly represented classes, confusing their class instances as overshoot values that appear in the dataset, and thus ignoring them.

This paper introduces the Enhancer, a new algorithm that improves the Cost-sensitive classification for Support Vector Machines, by multiplying in the training step the instances of the underrepresented classes. We have discovered that by oversampling the instances of the class of interest, we are helping the Support Vector Machine algorithm to overcome the soft margin. As an effect, it classifies better future instances of this class of interest.

Keywords: Learning Algorithms, Classification, Accuracy, Improvement, Unbalanced Datasets

1. INTRODUCTION

Most of the real-world data are unbalanced in terms of proportion of samples available for each class, which can cause problems such as over fit or little relevance. The Support Vector Machine (SVM), a classification technique based on statistical learning theory, was applied with great success in many challenging non-linear classification problems and was successfully applied to unbalanced data sets.

Proposed by Vapnik and his colleagues in 1990's [Vapnik, 2000], SVM is a new machine learning method based on Statistical Learning Theory and it is widely used in the area of regressive, pattern recognition and probability density estimation due to its simple structure and excellent learning performance. Joachims validated its outstanding performance in the area of text categorization in 1998 [Joachims, 1998]. SVM can also overcome the over fitting and under fitting problems [Hong et al., 2009], [Duan et al., 2009], and it has been used for unbalanced data classification [Li et al., 2009], [Xinfeng et al., 2009].

The SVM technique is based on two class classification. There are some methods used for classification in more than two classes. Looking at the two dimensional problem we actually want to find a line that "best" separates points in the positive class from the points in the negative class. The hyper plane is characterized by the decision function $f(x) = \text{sgn}(w, f(x) + b)$, where w is the weight vector, orthogonal to the hyper plane, b is a scalar that represents the margin of the hyper plane, x is the current sample tested, $f(x)$ is a function that transforms the input data into a higher dimensional feature space and "," representing the dot product. *Sgn* is the signum function. If w has unit length, then $< w, f(x) >$ is the length of $f(x)$ along the direction of w.

To construct the SVM classifier one has to minimize the norm of the weight vector w (where $\| w \|$ represents the Euclidian norm) under the constraint that the training patterns of each class reside on opposite sides of the separating surface. The training part of the algorithm needs to find the normal vector w that leads to the largest b of the hyper plane. Since the input vectors enter the dual only in form of dot products, the algorithm can be generalized to non-linear classification by mapping the input data into a higher-dimensional feature space via an a priori chosen non-linear mapping function f and construct a separating hyper plane with the maximum margin.

In solving the quadratic optimization problem of the linear SVM (i.e. when searching for a linear SVM in the new higher dimensional space), the training tuples appear only in the form of dot products, $< f(x_i), f(x_j) >$, where $f(x)$ is simply the nonlinear mapping function applied to transform the training tuples. Expensive calculation of dot products $< f(x_i), f(x_j) >$ in a high-dimensional space can be avoided by introducing a kernel function K:

$$K(x_i, x_j) = f(x_i) \cdot f(x_j) \quad (1)$$

The kernel trick can be applied since all feature vectors only occur in dot products. The weight vectors than become an expression in the feature space, and therefore f will be the

function through which we represent the input vector in the new space. Thus it is obtained the decision function having the following form:

$$f(x) = \text{sgn}(\sum_{i \in \Re} y_i a_i k(x, x_i) + b) \quad (2)$$

where a_i represent the Lagrange multipliers and the samples x_i for which $a_i > 0$ are called Support Vectors [Han & Kamber, 2006].

Because of the uneven distribution and the soft margin of the SVM, the algorithm tries to improve the general accuracy of classifying a dataset, and in this process it might misclassify a lot of weakly represented classes.

This paper introduces an algorithm named Enhancer aimed for increasing the TP of underrepresented classes of datasets, using Cost-sensitive classification and SVM.

2. COST-SENSITIVE APPROACH

In actual applications, it exist the problems that wrong classify result in different harm degree of different sort sample. The solution proposed in literature is the Cost-sensitive SVM approach [He & Garcia, 2009], [Dai et al., 2009], [Santos-Rodriguez et al., 2009], a new method for unbalanced classification.

Fundamental to the Cost-sensitive learning methodology is the concept of the cost matrix. This approach takes the classify cost into account, and it aims to reduce the classify cost to the least. Instead of creating balanced data distributions through different sampling strategies, Cost-sensitive learning targets the unbalanced learning problem by using different cost matrices that describe the costs for misclassifying any particular dataset. A very useful tool, the Confusion Matrix for two classes is shown in Table 1.

Table 1. Confusion Matrix for a two-class problem

		Predicted Class	
		Cls= 1	Cls= 0
Actual Class	Cls= 1	TP	FN
	Cls= 0	FP	TF

The true positives (TP) and true negatives (TN) are correct classifications. A false positive (FP) occurs when the outcome is incorrectly predicted as 1 (or positive) when it is actually 0 (negative). A false negative (FN) occurs when the outcome is incorrectly predicted as negative when it is actually positive.

In addition, the accuracy measure may be defined. It represents the ratio between correctly classified instances and the sum of all instances classified, both correct and incorrect ones. The above measure was defined as:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

More precisely, the classification gives equal importance to all the misclassified data (false negatives and false positives are equally significant). The Cost-sensitive classifications strive to minimize the total cost of the errors made by a misclassification, rather than the total amount of misclassified data.

Using the measures defined above, we calculated the accuracy mean, the true positives mean, and also the accuracy deviation and the true positives deviation:

$$AccMean = (\sum_{i=0}^{N\_TIMES} Acc_i) / N\_TIMES \quad (4)$$

$$TPMean = (\sum_{i=0}^{N\_TIMES} TP_i) / N\_TIMES \quad (5)$$

$$AccDeviation = \sqrt{\sum_{i=0}^{N\_TIMES} (AccMean - Acc_i)^2 / N\_TIMES} \quad (6)$$

$$TPDeviation = \sqrt{\sum_{i=0}^{N\_TIMES} (TPMean - TP_i)^2 / N\_TIMES} \quad (7)$$

3. DESCRIPTION OF THE ENHANCER

Experimentally we have found out that the features that help in raising the TP of a class are the cost matrix and the amount of instances that the class has. The last one can be modified by multiplying the number of instances of that class that the dataset initially has.

The algorithm proposed for increasing the TP of weakly represented classes, the Enhancer is detailed in the following pseudo code:

1. *Read and validate input;*
2. *For all the classes that are not well represented:*

BEGIN
Evaluate class with no attribute added
Evaluate class at Max multiplication rate
Evaluate the class at Half multiplication
REPEAT
Flag = False
Evaluate the intervals (beginning, middle), (middle, end)
If the end condition is met
(i.e. If the difference between the beginning and the end of an interval is very small, under a set epsilon AND
If $|\Delta TP_i + \Delta Acc| \geq m$,
where $m = (AccDeviation + TPDeviation_i)$ *)*
Flag = True
If the first interval has better results we should use this, otherwise the other

Find the class evaluation after multiplying class instances middle times
UNTIL Flag = False
END

3. *Multiply all the classes with the best factor obtained;*
4. *Evaluate dataset.*

While *reading and validating the input* we collected from the command line the parameters that were used by this classifier, together with the classifier parameters that were usually transmitted to the program. The input parameters needed were the number of the class that needs to have its TP improved and the ε that is the maximum allowed difference between the evaluation of the two intervals (*beginning, middle*) and (*middle, end*).

Our classifier had also as input parameters the multiplicands that the optimization algorithm had used. There are available two kinds of evaluations that also accept class multiplication:

• Evaluating a dataset with only the instances of one class being multiplied, and keeping the other still to their initial value. This kind of operation was especially useful when we tried to find out what was the best multiplicand for a certain class.

• Evaluation of a dataset where the instances of all classes could undergo a multiplication process. The multiplication of the classes could be any real number greater or equal to 1. If the multiplicand was 1, then the class remained with the initial number of instance.

One of the most important parts in this pseudo code is knowing what, when, and how to evaluate data set, in order to maximize efficiency of the algorithm. This problem only appears when the search for the perfect number to be used as a multiplier for a certain class is not assisted by the human component.

It is also important to avoid performing the evaluation on data that the algorithm used to train the model on, because otherwise the algorithm is going to over fit on this particular dataset, and when new data is going to be introduced to be tested, the results are going to be disastrous. This way of evaluation is the 10 fold cross validation. Like this the dataset is being randomized, and stratified using an integer seed that takes values in the range 1-10. The algorithm performs 10 times the evaluation of the data set, and all the time has a different test set (Fig. 1).

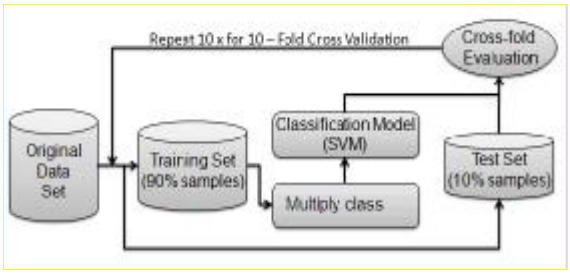

Fig. 1. 10 fold cross validation

So, after performing the stratification, each time the data set was split into the training and test set, the Enhancer took the training set and applied *classMultiply()* on it. Like this the instances that were going to be multiplied were not going to be among that data that was going to test the result of the SMO model, the Weka implementation of SVM. The performance of the algorithm is only due to the multiplied data, and there is no over fitting to this specific data set. The data was trained in order to be evaluated as accurately as possible by a general test set, and not only by the one for testing.

The instances were multiplied using the properties of the *Instances* object in which they were stored following this pseudo code:

1 aux← all instances of class x from dataset
2 for i=0 to max do
3 add (instance from aux to dataset)
4 Randomize dataset

By performing this series of operations the number of instances of the desired class was multiplied by the desired amount and in the same time we had a good distribution of instances inside the dataset in order not to harm or benefit any of the classes in the new dataset.

In order to see what the best improvement is, we need to calculate an ending property of the logarithm. After some experiments the conclusion was that we must optimize the TP and in the same time keep the accuracy as high as possible. This can be translated as follows:

$$j = \Delta TP_i + \Delta A_{CC} = \max \tag{8}$$

This means that we are trying all the time to maximize the TP of classes and also the Accuracy. The only flaw in this equation is the Accuracy is medium (50%) and the TP of that certain class is really close to 0. If realize to get the TP of the class as high as 80-90%, the loss in the accuracy, that is going to appear inevitably, is going to pass unnoticed by this function. That is why we needed to introduce the following constraint: $\Delta A_{CC} > q$, where q is the minimum allowed drop in the accuracy.

The Enhancer algorithm described in the pseudo code used a *Divide et Impera* technique, that searched in the space (0 multiplication – max multiplication) for the optimal multiplier for the class. The algorithm is going to stop its search under two circumstances:

• The granulation is getting to thin, i.e., the difference between the beginning and end of an interval is very small (under a set epsilon). This constraint is set, in order not to let the algorithm wonder around searching for solutions that vary one from another by a very small number ($<10^{-2}$).

• The modulus of the difference between the $\Delta TP_i + \Delta A_{CC}$ from the first and the second interval should be bigger that a known value. This value is the considered to be the deviation of the Accuracy added to the deviation of the TP of that class:

$$m = sA_{CC} + sTP \quad (9)$$

After finding the best multiplicand for the class that we are trying to optimize, we constructed a training set that contained each class instances multiplied by the optimal multiplicand found at the previous step. A fine tuning was performed on the multiplicands of each of the other weakly represented classes, in order to raise the accuracy and the TP of the other classes while keeping the TP of the interested class at least at the same value that the algorithm retrieved.

4. EXPERIMENTAL RESULTS

The classification is unbalanced when data presents many examples from one class and few from the other class, and the less representative class is the one which has more interest [Garcia et al., 2009].

For the evaluation, we used three unbalanced data sets, the Cleveland, the Dermatology, and the Blood Transfusion data-sets obtained from the online UCI Machine Learning Repository. A brief description of these datasets is presented in Table 2. The datasets have accuracy levels which are not very high, so improvement is possible.

Table 2. The datasets used in the experiments

Dataset	No. of attributes	No. of instances	Attributes types
Cleveland	13+1	303	Num, Nom
Dermatology	34+1	358	Num, Nom
Blood Transfusion	4+1	748	Num, Nom

The class distribution for the datasets is illustrated below (Fig. 2, Fig. 3, and Fig. 4):

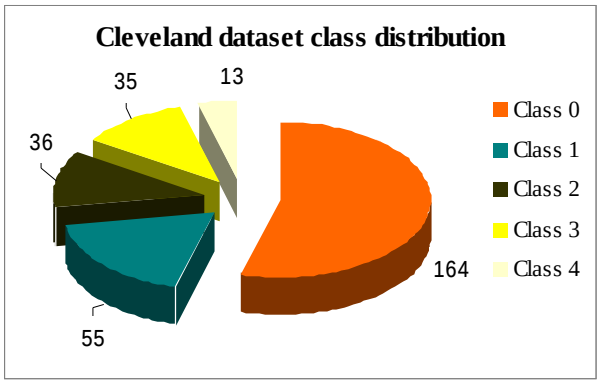

Fig. 2. Cleveland dataset class distribution

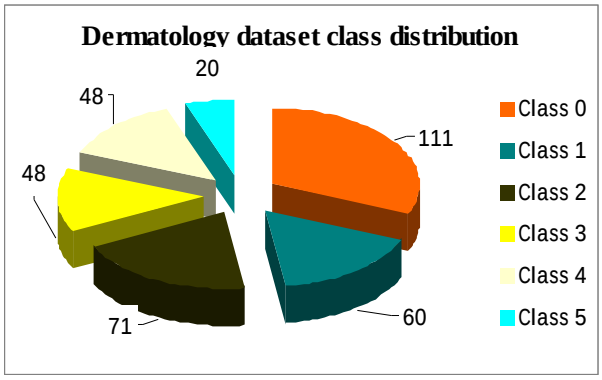

Fig. 3. Dermatology dataset class distribution

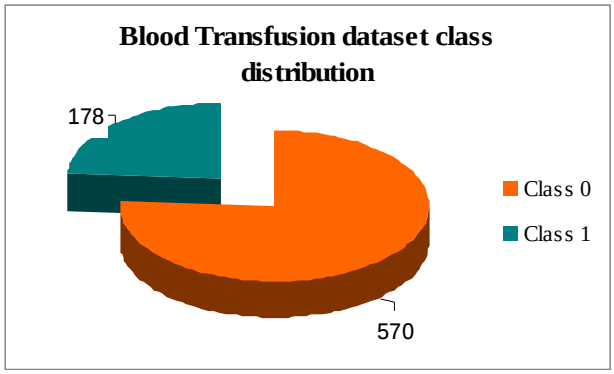

Fig. 4. Blood Transfusion dataset class distribution

In order to improve the classification of the weakly represented classes in those datasets, in which they are in very small numbers with respect to the other classes, two approaches were tested:

- Cost-sensitive classification;
- Multiplication of the instances of weakly represented classes.

4.1 Cost-sensitive classification

In the case of the Cost-sensitive classification, the main aim was to find a good cost matrix, to increase the cost of wrongly classified instances that belong to the weakly represented classes. In order to perform this, we used the Cost-Sensitive Classifier that can be found in *Weka.classifiers.meta* on the three datasets described above as follows:

- We have set as cost matrix the default cost matrix (0 on the main diagonal and 1 in rest);
- We evaluated the datasets;
- We "fixed" the cost matrix, to increase the cost of the wrongly classified instances where the Confusion Matrix indicated FN, to force the algorithm to correctly classify those instances as well.

• We re-evaluated the datasets and redid the previous step.

By modifying the cost matrix, we obtained a variation quite high in the TP of Class 0 (88%-97%, Fig. 5). This class has 164 instances from the total of 303 that are available in the dataset.

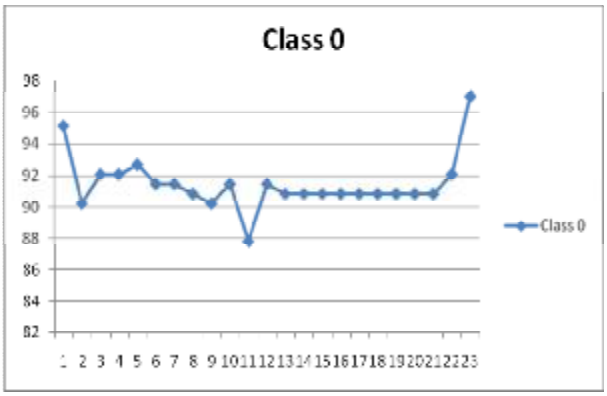

Fig. 5. Class 0 TP variation with respect to Cost Matrix change

Unfortunately, this increase in the TP of the most representative class affected the other, weaker represented classes with a low TP (Fig. 6, Fig. 7, and Fig. 8).

The instances of the Classes 0 and 1 aren't so well separated, because they have a high value for the FP and FN. In the case of class 1, that has 55 instances (18% of the total amount), we observed that its TP never goes higher than 32%.

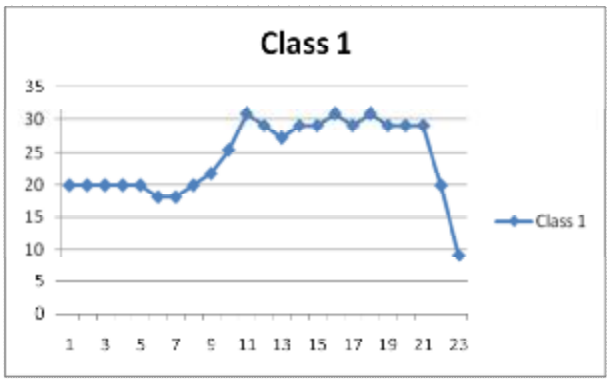

Fig. 6. Class 1 TP variation with respect to Cost Matrix change

Class 2 has 36 instances, which represent 11% of the full dataset. In the case of class 2 we observed an ever tighter connection with another class, unlike the connection between the classes 0-1. This class is strongly connected with class 1. The TP of the two classes evolve almost complementary one from another (when one TP rises, the other falls and the other way around). The TP of class 2 also seemed to be bounded by about 30%.

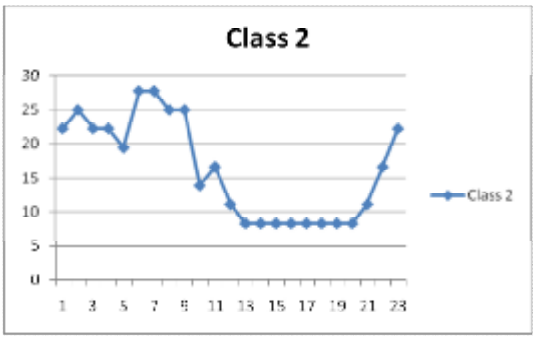

Fig. 7. Class 2 TP variation with respect to Cost Matrix change

Class 3 has 35 instances, meaning 11% of the dataset. This class showed the most spectacular evolution of all. By modifying the cost matrix we were able to change the level of the accuracy from 20% to 60%.

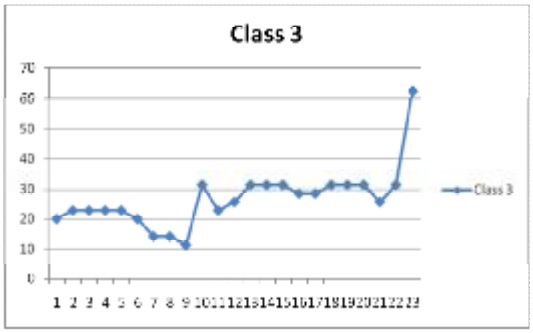

Fig. 8. Class 3 TP variation with respect to Cost Matrix change

Class 4, which has 13 instances, meaning 4% from the total amount of instances is the worst class that the Cost-Sensitive Classifier managed to classify. The TP of this class was also bounded by a very low 25%. This class seemed to be very weekly represented for the SVM to classify it correctly, and regardless the value of the cost of a misclassification it kept on making the same confusions over and over again (Fig. 9).

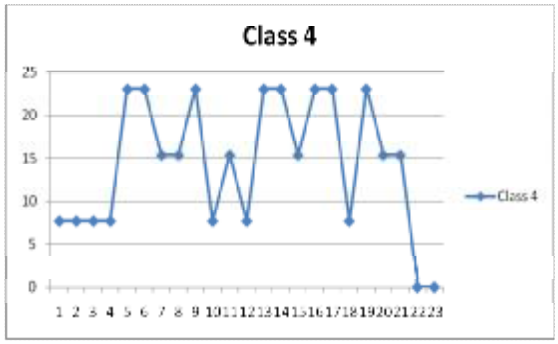

Fig. 9. Class 4 TP variation with respect to Cost Matrix change

Unfortunately, the increase in the TP of the Class 0 of Blood Transfusion dataset affected the other one, with a low TP. This class has 570 instances from the total of 748 that are available in the dataset (Fig. 10).

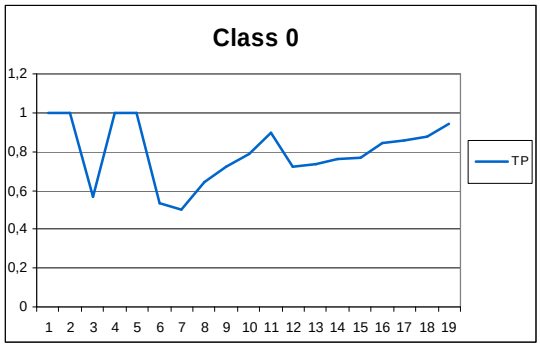

Fig. 10. Class 0 TP variation of Blood Transfusion dataset with respect to Cost Matrix change

By modifying the cost matrix, we obtained a high variation in the TP of Class 1. In the case of Class 1, that has 178 instances (24% of the total amount), we observed that its TP never goes higher than 81.5% (Fig. 11).

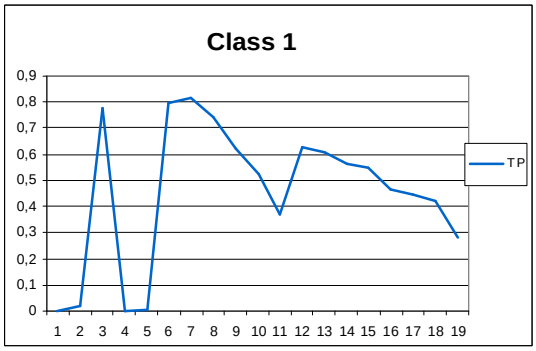

Fig. 11. Class 1 TP variation of Blood Transfusion dataset with respect to Cost Matrix change

In the case of Dermatology dataset, the accuracy and the TP of the classes 0, 2, 4 and 5 remained constant with respect to Cost Matrix change (Fig. 12).

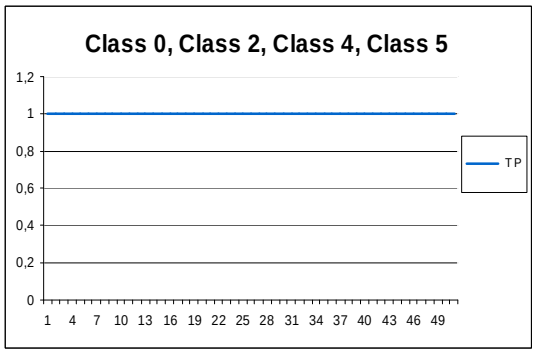

Fig. 12. Class 0, Class 2, Class 4 and Class 5 TP variation of Dermatology dataset with respect to Cost Matrix change

The TP of the Classes 1 and 3 evolved almost complementary one from another (when one TP rises, the other falls). We mention that the Class 1 has 60 instances, which represent 17% of the full dataset and the Class 3 has 48 instances (13% of the total amount) (Fig. 13 and Fig. 14).

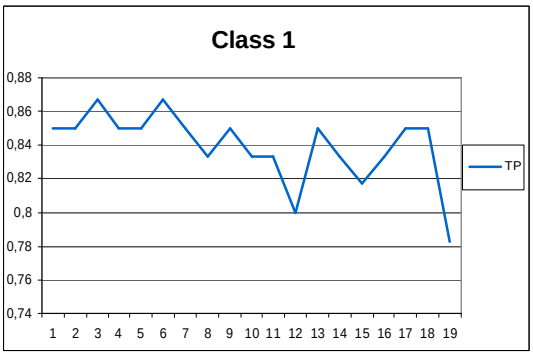

Fig. 13. Class 1 TP variation of Dermatology data set with respect to Cost Matrix change

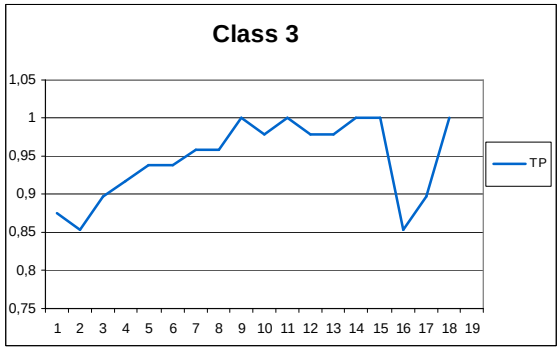

Fig. 14. Class 3 TP variation of Dermatology data set with respect to Cost Matrix change

After the experiments, we have found out that most of the time, when we were dealing with the classification of unbalanced data sets, and trying to improve the accuracy with which the classes were predicted, two main things happened:

• The correct classifications of certain classes' increased, and in the same time the general accuracy rose. In this case, the growth of the TP of that class hadn't influenced the accuracy of other classes, so more instances were correctly classified, thus the increase in the accuracy.

• The correct classification of certain classes' increased and the general accuracy remained the same. In this case, the rise of the TP of a class affected the evaluation of other classes, which were going to get sloppier.

Cost-sensitive classification proved to be a good method of improving the TP of the unbalanced classes in the dataset that were weakly connected with one-another.

When the instances of certain classes were not correctly identified, this could be because of the soft margin of the SVMs, which were interpreted that the instances of the

weakly represented classes are just few errors in the classification of the larger classes.

4.2 Multiplying underrepresented classes

In order to improve the classification of one class of interest from the training dataset using SVM, we needed to improve its chances of being recognized. In order to recognize classes, SVM needed support vectors from those classes and that's why we had increased the number of instances of a weakly represented class and in the same time we had kept the value of the other classes constant. First, we tried to find out what is the best multiplier to use for the several classes and how much did it affect the rest of the evaluation.

After applying the class multiplications all the TP of Class 1 hits a zone of instability, until the multiplying factor reached 0.9, when the TP ascent stabilized. In the end, the TP of this class reached almost 60%, which is double from the accuracy that was reached using only the Cost-sensitive classification. Seemingly the problem in this case is that the accuracy was dropping or remaining constant, while the TP of the class was getting bigger and bigger (Fig. 15).

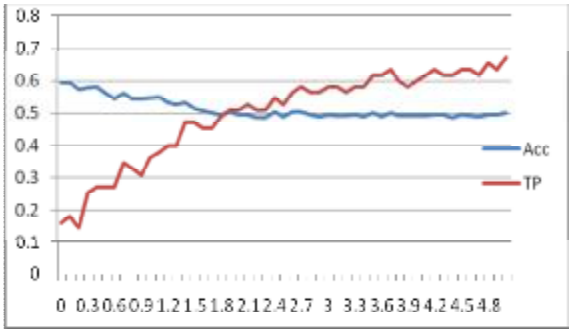

Fig. 15. The evolution of the TP of Class 1 and the general accuracy with respect to the number of instances of Class 1

As in the case of Class 1, the algorithm provided very good results in the classification of Class 2. Everything said about the ascent of the TP and the standing still of the accuracy remains valid in this case also. We observed that where we used the Cost-sensitive classification only, the increase in the TP of the class is more than double (Fig. 16).

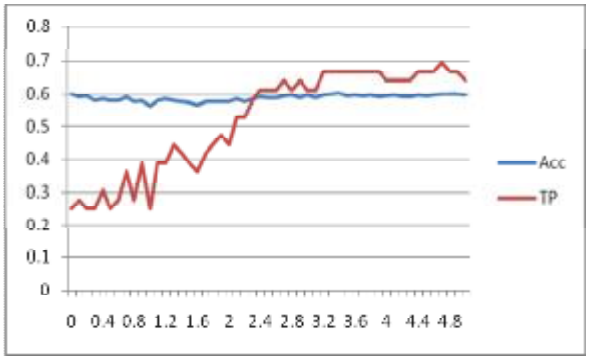

Fig. 16. The evolution of the TP of Class 2 and the general accuracy with respect to the number of instances of Class 2

This algorithm helps Class number 3 as well. Although it started from a quite high, the TP of the class improved more and stabilized its ascent at around 0.88 (Fig. 17).

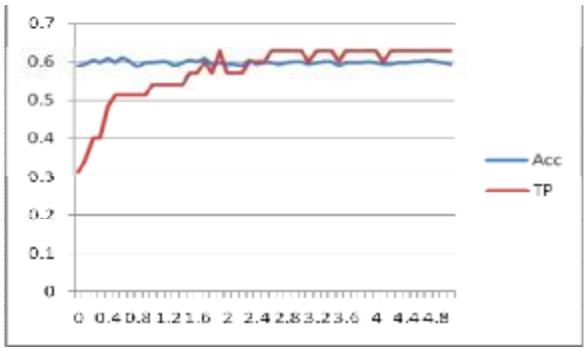

Fig. 17. The evolution of the TP of Class 3 and the general accuracy with respect to the number of instances of Class 3

The improving of the TP of Class 4 seemed to be very unstable, with peeks of maxima, which couldn't be taken into consideration. The number of instances that this class has, is far too few (4%) and even though we multiplied the number of instance that belong to the class, the dataset didn't know how to construct the boundary hyper plane, as the special span of the class distribution was too narrow (Fig. 18).

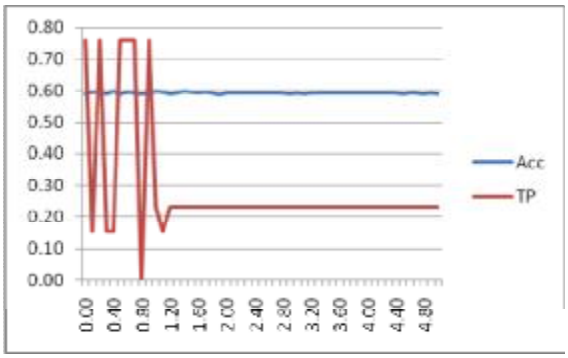

Fig. 18. The evolution of the TP of Class 4 and the general accuracy with respect to the number of instances of Class 4

This algorithm helps the Class 3 of the Dermatology dataset as well. Although it started from a quite high (0.85), the TP of the class stabilized its ascent at around 0.89 (Fig. 19).

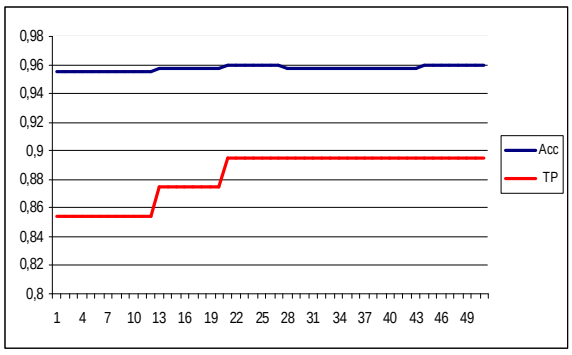

Fig. 19. The evolution of the TP of Class 3 and the general accuracy with respect to the number of instances of Class 3 (Dermatology dataset)

The accuracy of the Class number 1 reached 0.95, meaning a better value than the one reached using only the Cost-sensitive classification (Fig. 20).

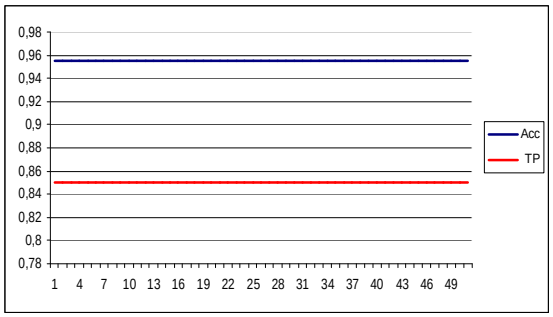

Fig. 20. The evolution of the TP of Class 1and the general accuracy with respect to the number of instances of Class 1 (Dermatology dataset)

The accuracy and the TP of the Classes 0, 2, 4 and 5 remained constant with respect to Cost Matrix change (Fig. 21).

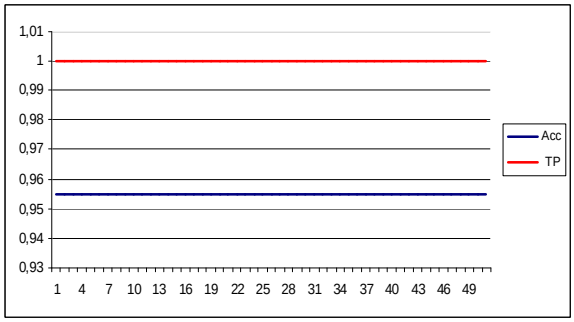

Fig. 21. The evolution of the TP of Classes 0, 2, 4 and 5 and the general accuracy with respect to the number of instances of Classes 0, 2, 4, and 5 (Dermatology dataset)

After applying the class multiplications, all the TP of Class 1 (in the case of Blood Transfusion dataset) hit some zones of instability, until the multiplying factor reached 0.8, when the TP ascent stabilized and grew to 0.9. Seemingly the problem in this case is that the accuracy was dropping or remaining constant, while the TP of the Class was getting bigger (Fig. 22).

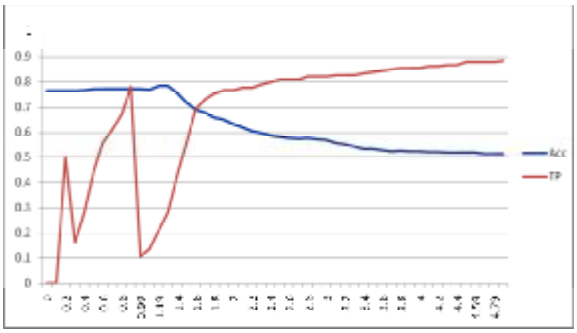

Fig. 22. The evolution of the TP of Class 1 (Blood Transfusion dataset) and the general accuracy with respect to the number of instances of Class 1

So, the Enhancer multiplied the information accordingly, such that to maximize $\Delta TP_i + \Delta A_{CC}$, so the accuracy does not fall below a set ε. We set ε to 0.05 (5%) and we concluded that with the new algorithm, the TP of certain classes of interest were increased significantly while keeping the general accuracy in the desired range (Fig. 23, Fig. 24, and Fig. 25).

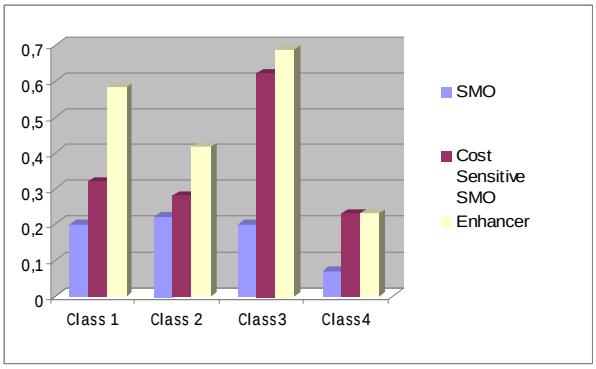

Fig. 23. Comparison between the TP of the classes resulting Cost-sensitive SMO Evaluation and with the Enhancer (Cleveland dataset)

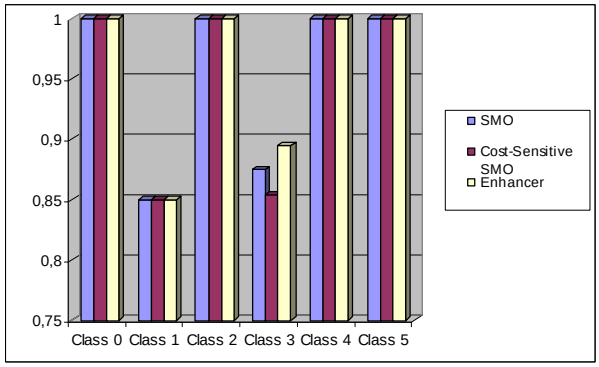

Fig. 24. Comparison between the TP of the classes resulting Cost-sensitive SMO Evaluation and with the Enhancer (Dermatology dataset)

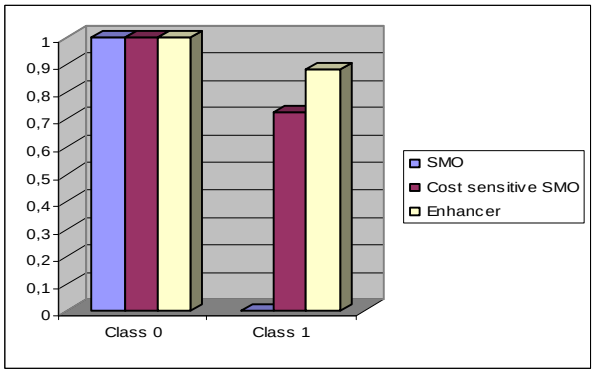

Fig. 25. Comparison between the TP of the classes resulting Cost-sensitive SMO Evaluation and with the Enhancer (Blood Transfusion dataset)

We observed that the last classifier performs the best and the Enhancer algorithm could have pointed even more accurately

the instances that belong to the class of interest, but with the downside of pulling the general accuracy below the threshold preset ε.

The cost matrices that were used here are the best ones that were found in the evaluation step. The rows are read as "classified as", and columns as "actual class" (Table 3, Table 4, Table 5).

Table 3. Cleveland cost matrix

Cls 0	Cls 1	Cls 2	Cls 3	Cls 4	
0.0	9.5	75.6	105.8	116.2	Cls 0
28.4	0.0	14.2	23.6	86.9	Cls 1
1.9	120.0	0.0	66.1	18.9	Cls 2
95.4	98.3	115.3	0.0	109.6	Cls 3
70.9	21.7	27.4	67.1	0.0	Cls 4

Table 4. Dermatology cost matrix

Cls 0	Cls 1	Cls 2	Cls 3	Cls 4	Cls 5	
0.0	1.0	1.0	1.0	1.0	1.0	Cls 0
1.0	0.0	1.0	210.0	1.0	1.0	Cls 1
1.0	1.0	0.0	1.0	1.0	1.0	Cls 2
1.0	343.0	1.0	0.0	1.0	1.0	Cls 3
1.0	1.0	1.0	1.0	0.0	1.0	Cls 4
1.0	1.0	1.0	1.0	1.0	0.0	Cls 5

Table 5. Blood Transfusion cost matrix

Cls 0	Cls 1	
0.0	12.0	Cls 0
30.0	0.0	Cls 1

We also noted that the Enhancer had highest values for TP instances among all other metaclassifiers while the accuracy was kept at an acceptable level (Table 6, Table 7, and Table 8).

Table 6. The results obtained by compared metaclassifiers on Cleveland dataset

MetaClassifier with SMO	Acc (%)	TP				
		Cls 0	Cls 1	Cls 2	Cls 3	Cls 4
AdaBoost	59.07	0.95	0.18	0.17	0.23	0.00
Attribute Selected	57.42	0.92	0.16	0.14	0.26	0.08
CVParameter Selection	59.07	0.95	0.20	0.17	0.20	0.00
Dagging	59.40	0.95	0.27	0.03	0.26	0.00
Decorate	58.08	0.95	0.15	0.17	0.20	0.00
Ensemble Selection	57.09	0.95	0.11	0.11	0.17	0.08
Filtered Classifier	58.08	0.92	0.11	0.22	0.29	0.08
MetaCost	58.08	0.95	0.2	0.11	0.20	0.00
MultiClass Classifier	54.12	1.00	0.00	0.00	0.00	0.00
MultiBoost AB	50.82	0.86	0.18	0.06	0.03	0.00
CostSensitive Classifier	59.07	0.95	0.32	0.28	0.62	0.23
Enhancer	**57.20**	**0.95**	**0.58**	**0.42**	**0.68**	**0.23**

Table 7. The results obtained by compared metaclassifiers on Dermatology dataset

Meta Classifier with SMO	Acc (%)	TP					
		Cls 0	Cls 1	Cls 2	Cls 3	Cls 4	Cls 5
Ada Boost	95.81	1.00	0.85	1.00	0.88	1.00	1.00
Attribute Selected	97.49	1.00	0.95	0.97	0.90	1.00	1.00
CVParameter Selection	95.81	1.00	0.85	1.00	0.88	1.00	1.00
Dagging	94.69	1.00	0.92	1.00	0.89	1.00	0.55
Decorate	96.37	1.00	0.88	1.00	0.88	1.00	1.00
Ensemble Selection	94.41	0.97	0.93	1.00	0.85	0.92	0.90
Filtered Class	96.09	1.00	0.87	1.00	0.88	1.00	1.00

ifier							
Meta Cost	68.44	0.99	0.00	1.00	0.00	0.96	0.90
Multi Class Classifier	96.37	1.00	0.90	1.00	0.88	1.00	0.95
Multi Boost AB	96.09	1.00	0.87	1.00	0.88	1.00	1.00
CostSensitive Classifier	95.53	1.00	0.85	1.00	0.85	1.00	1.00
Enhancer	**95.50**	**1.00**	**0.90**	**1.00**	**0.90**	**1.00**	**1.00**

Table 8. The results obtained by compared metaclassifiers on Blood Transfusion dataset

MetaClassifier with SMO	Accuracy (%)	TP	
		Cls 0	Cls 1
AdaBoost	77.27	0.97	0.14
Attribute Selected	76.20	1.00	0.00
CVParameter Selection	76.20	1.00	0.00
Dagging	76.20	1.00	0.00
Decorate	76.47	0.99	0.03
Ensemble Selection	77.01	0.92	0.28
Filtered Classifier	76.20	1.00	0.00
MetaCost	76.20	1.00	0.00
MultiClass Classifier	76.20	1.00	0.00
MultiBoost AB	76.34	0.99	0.01
CostSensitive Classifier	70.05	0.73	0.62
Enhancer	**51.30**	**1.00**	**0.88**

5. CONCLUSIONS

This paper is focused on providing the Enhancer, a viable algorithm for improving the SVM classification of unbalanced datasets.

Most of the times, in unbalanced data sets, the classifiers have a tendency of classifying in a very accurate manner the instances belonging to the best represented classes and do a sloppy job with the rest. In order to overcome this problem we have developed the new classifying algorithm that can classify the instances of a class of interest better than the classification of the usual SVM algorithm. All of this is happening while keeping the accuracy at an acceptable level.

The algorithm improves the classification of the weakly represented classes in the dataset, with the condition that the class must have more than 5% of the total number of instances. For a smaller number of instances, the SVM just over fits the built model to the few amounts of data, and produces no real results when faced with new test instances. The idea of multiplying the unrepresented classes is original and came from the experimental work. We have also discovered that by over sampling the instances of the class of interest, we are helping the SVM algorithm to overcome the soft margin. As an effect, it classifies better future instances of this class of interest.

The algorithm improves the classification of the weakly represented class in the dataset and it can be used for solving real medical diagnosis problems. This solution is especially important when it is far more important to classify the instances of a class correctly, and if in this process we might classify some of the other instances as belonging to this class we do not produce any harm. For instance it is better to send people suspect of different diseases to further investigations, than sending ill people at home and telling them they don't have anything to worry about.

As a future work, we propose to maximize accuracy with geometric mean metric in order to balance both classes at the same time. This evaluation measure will allow us to simultaneously maximize the accuracy in positive and negative examples with a favourable trade-off.

REFERENCES

Dai, Y., Chen, H., and Peng, T. (2009). Cost-sensitive Support Vector Machine based on weighted attribute, *2009 International Forum on Information Technology and Applications*, pp. 690-692, 15-17, May, 2009, Chengdu, China.

Duan, X., Shan, G., and Zhang, Q. (2009). Design of a two layers Support Vector Machine for classification, *2009 Second International Conference on Information and Computing Science*, pp. 247-250, May, 21-22, 2009, Manchester, UK.

Garcia, S., Fernandez, A., and Herrera, F. (2009). Enhancing the effectiveness and interpretability of decision tree and

rule induction classifiers with evolutionary training set selection over imbalanced problems, *Applied Soft Computing 9 (2009)*, 1304–1314, Elsevier, 2009.

Han J. and Kamber, M. (2006). Data Mining*: Concepts and Techniques, Second Edition*, Morgan Kaufmann Press, Elsevier Inc, San Francisco, 2006, pp. 337.

He, H. and Garcia, E. A. (2009). Learning from imbalanced data, *IEEE Transactions on Knowledge and Data Engineering*, VOL. 21, NO. 9, September, 2009.

Hong, M., Yanchun, G., Yujie, W., and Xiaoying, L. (2009). Study on classification method based on Support Vector Machine, *2009 First International Workshop on Education Technology and Computer Science*, pp.369-373, March, 7-8, 2009, Wuhan, China.

Joachims, I. (1998). Text categorization with Support Vector Machines: Learning with many relevant features, *Proceedings of the European Conference on Machine Learning*, Berlin: Springer, 1998.

Li, Y., Danrui, X., and Zhe, D. (2009). A new method of Support Vector Machine for class imbalance problem, *2009 International Joint Conference on Computational Sciences and Optimization*, pp. 904-907, April 24-26, 2009, Hainan Island, China.

Santos-Rodriguez, R., Garcia-Garcia, D., and Cid-Sueiro, J. (2009). Cost-sensitive classification based on Bregman divergences for medical diagnosis, *2009 International Conference on Machine Learning and Applications*, pp. 551-556, 13-15, December, 2009, Florida, USA.

Vapnik, V N. (2000). *The nature of statistical learning theory*, New York: Springer-Verlag, 2000.

Xinfeng, Z., Xiaozhao, X., Yiheng, C., and Yaowei, L. (2009). A weighted hyper-sphere SVM, *2009 Fifth International Conference on Natural Computation*, pp. 574-577, 14-16, August, 2009, Tianjin, China.

*** (2010) University of California Irvine, UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/.