Improving Fault Diagnosis for Wireless Sensor Network's Data using Support Vector Machines

Maria Muntean*, Honoriu Vălean**

**1 Decembrie 1918 University of Alba Iulia, Romania*
(e-mail: mmuntean@ uab.ro)
***Technical 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. The experiments were performed using real data acquired from a monitoring sensor system, stored in three databases and replicated on an aggregation server.

Keywords: Learning Algorithms, Classification, Accuracy, Improvement, Wireless Sensor Network

1. INTRODUCTION

Real-time monitoring data mining has been a necessary means of improving operational efficiency, economic safety and fault diagnosis of Wireless Sensor Network's data. Along with the rapid development of sensors and detection technique and abundant signal sources, more and more data are accumulated which provides a basis for fault diagnosis for big machines.

Fault diagnosis is a pattern recognition process essentially. First of all, the signal parameter is obtained in original space, which is mapped into observation space to extract feature vectors subsequently, the fault feature database is established, and these vectors are input into classifier for fault diagnosis (Yuan, 2011; Hong and Jing, 2011).

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 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 $f(x) = \mathrm{sgn}(w, \phi(x) + b)$, 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, $\phi(x)$ 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 $< w, \phi(x) >$ is the length of $\phi(x)$ 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 ϕ 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, $< \phi(x_i), \phi(x_j) >$, where $\phi(x)$ is simply the nonlinear mapping function applied to transform the training tuples. Expensive calculation of dot products $< \phi(x_i), \phi(x_j) >$ in a high-dimensional space can be avoided by introducing a kernel function *K*:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \tag{1}$$

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 ϕ 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:

$$f(x) = \operatorname{sgn}\Big(\sum_{i \in \Re} y_i \alpha_i k(x, x_i) + b\Big) \tag{2}$$

where α_i represent the Lagrange multipliers and the samples x_i for which $\alpha_i > 0$ are called Support Vectors (Han and 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 and 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.

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.

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:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

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:

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

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

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

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

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 $|\Delta TP_i + \Delta Acc| \geq \mu$,
where $\mu = (AccDeviation + TPDeviation_i)$ )
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**).

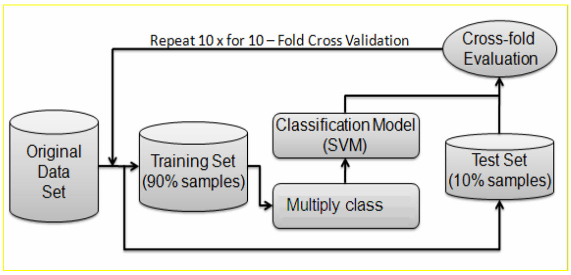
Repeat 10 x for 10 – Fold Cross Validation
Cross-fold Evaluation
Original Data Set
Training Set (90% samples)
Classification Model (SVM)
Multiply class
Test Set (10% samples)

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:

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

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: $\Delta A_{CC} > \theta$, where θ 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:

$$\mu = \sigma A_{CC} + \sigma TP \tag{9}$$

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

4.1. The autonomous measuring system description

Classification of sensory data is a major research problem in wireless sensor networks and it can be widely used in reducing the data transmission in wireless sensor networks effectively and also in process monitoring.

In our wind energy monitoring, sensor node monitors six attributes: speed, direction, temperature, pressure, humidity, and battery voltage.

The autonomous measuring system used for the estimation of wind energy is composed by:

- a measurement tower 85 meters height, with assembly and structure accessories (guy wires, anchors, clamping fixture, auxiliary masts, mounting plate, aviation light and/or beaconing) (**Fig.** 2);
- four Thies first class wind speed sensors (V1, V2, V3, V4, depending upon climatic conditions with heating);
- two Vilmers wind direction sensors D1 respectively D2;
- two temperature sensors with shielding T1, T2;
- one barometric pressure sensor P1;
- one relative humidity sensor H Energy Supply System consist on PV-Module, Voltage-regulator, 12V Battery, in lockable steel cabinet;
- wireless data transmission system (GSM-Modem + 1xCampbell CR1000 data logger inclusive leads).

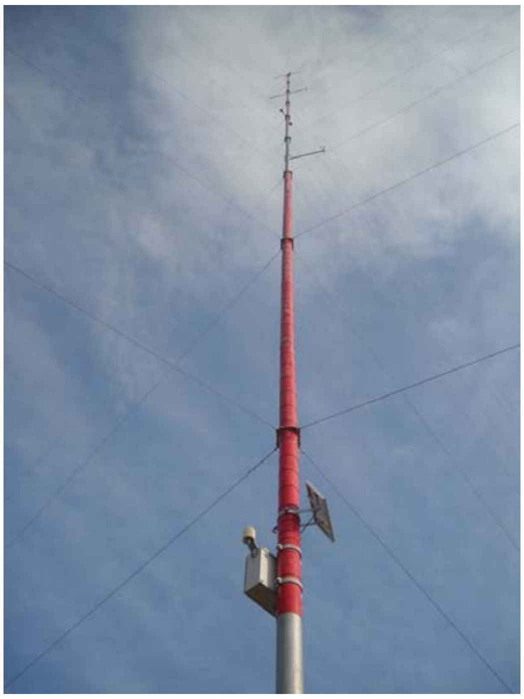

Fig. 2. The measurement tower

The sensors are distributed on the entire measurement tower (**Fig.** 3).

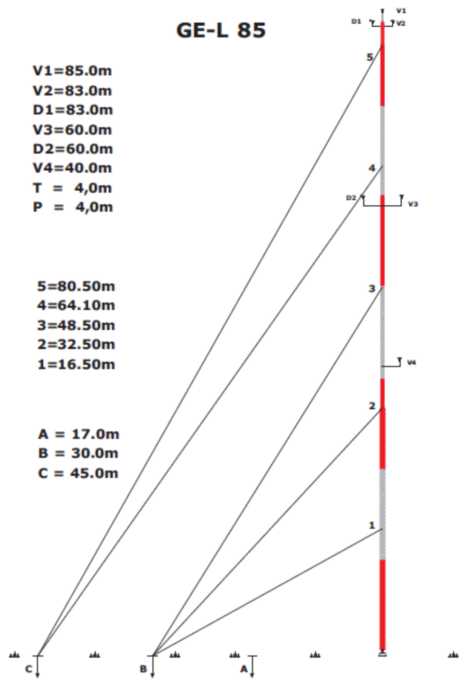

Fig. 3. Sensors distribution on the measurement tower

The wind speed sensors are designed for the measurement of the horizontal component of the wind speed in the fields of meteorology and environmental protection and the wind direction transmitter serves for the detection of the horizontal wind direction. Some special characteristics are: high level of measuring accuracy and resolution, high damping ratio at a small delay distance, low starting threshold and magnetic coupling, which is free of wear. The temperature and pressure sensors measure the temperature and relative humidity of the ambient air. A radiation shield protects the sensor against rain and solar radiation. The humidity sensor is design for measurement of the humidity of the air. The U_Batt sensor measures the battery voltage values, and PTemp_C registers the temperature from the lockable steel cabinet (Fig. 4).

Fig. 4. The lockable steel cabinet

4.2 The distributed database replication

Our distributed database system includes three servers situated on three geographic remote sites. Each server has a client with insert rights (Fig. 5).

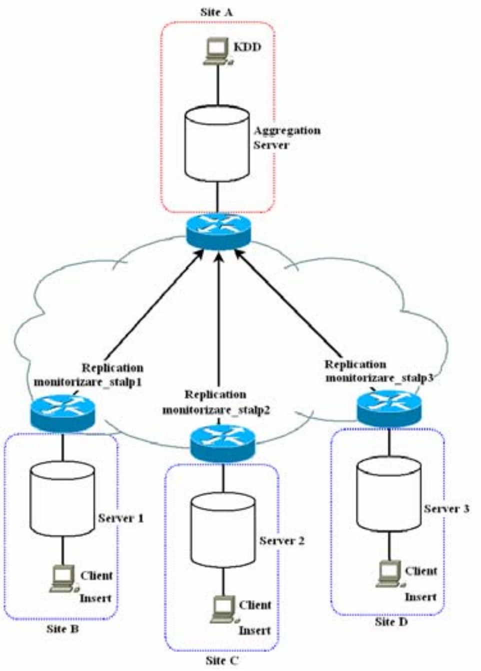

Fig. 5. The distributed database replication schema

All three databases contain the information acquired from the sensors and are replicated on an aggregation server. This server has a client with select rights. On this machine is running our proposed algorithm that detects the anomaly values from the aggregated database.

4.3 The database description

The experiments performed in this paper evaluate data obtained from our wireless sensor network and accumulated in the period 1-31 May 2010.

The diagram of one database and a brief description of the aggregated dataset are presented in Fig. 6 and Table 2.

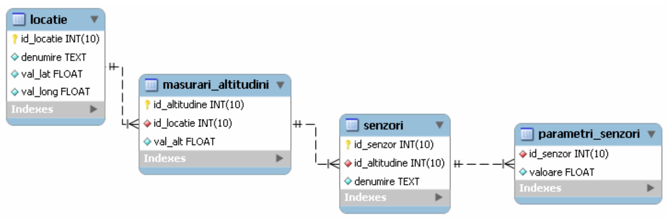

Fig. 6. The structure of one database

Table 2. The dataset used in the experiments

Dataset	No. of attributes	No. of instances	Attributes types
Aggregated dataset	4+1	17280	Num, Nom

The class distribution for the dataset is illustrated below (Fig. 7):

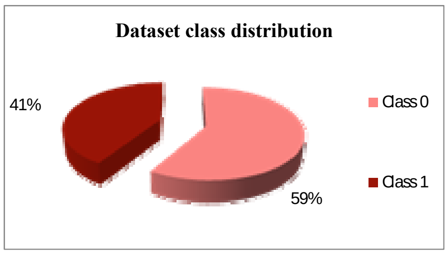

Fig. 7. The aggregated dataset class distribution

In order to improve the classification of the weakly represented class in this dataset, 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.

4.4 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 class, in our case to the anomaly class. In order to perform this, we used the Cost-Sensitive Classifier that can be found in *Weka.classifiers.meta* on the dataset 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 dataset;
- 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 dataset and redid the previous step.

By modifying the Cost Matrix, we obtained a variation quite high in the TP of Class 1 (60%-92%, **Fig. 8**). The accuracy of classification took values between 84% and 93%. This class has 7128 instances from the total of 17280 that are available in the dataset.

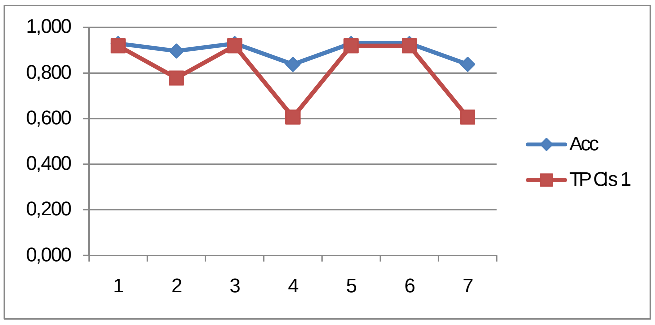

Fig. 8. Class 1 TP variation with respect to Cost Matrix change

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.

4.5 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 anomaly class 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 1.0, when the TP ascent stabilized. The accuracy of the Class 1 reached 0.91 for the TP value equal to 100%, meaning a comparable value as the ones obtained using only the Cost-sensitive classification (Fig. 9).

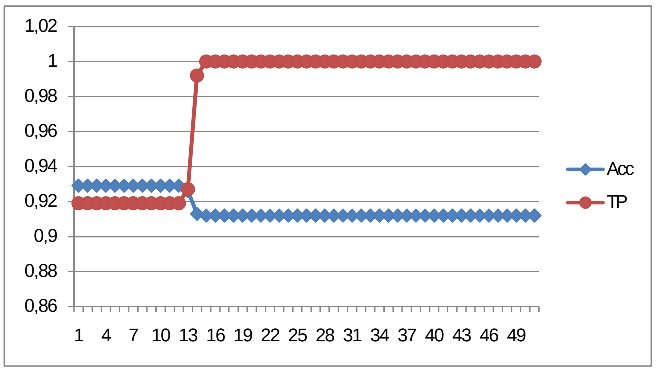

Fig. 9. The evolution of the TP of Class 1 and the general accuracy with respect to the number of instances of Class 1

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 a certain class of interest was increased significantly while keeping the general accuracy in the desired range (Fig. 10).

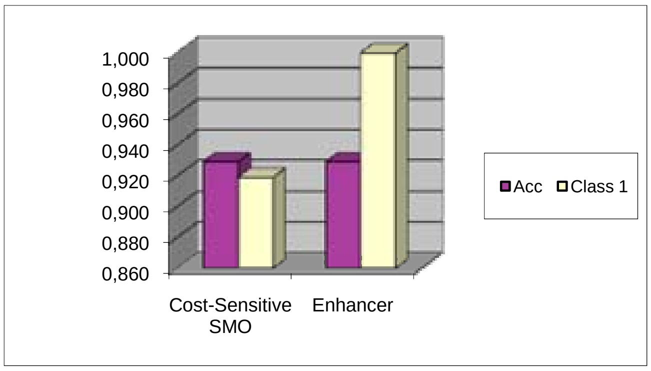

Fig. 10. Comparison between the TP of the class 1 resulting Cost-sensitive SMO Evaluation and with the Enhancer

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 was used here is the best one found in the evaluation step. The rows are read as "classified as", and columns as "actual class" (Table 3).

Table 3. The cost matrix used

Cls 0	Cls 1	
0.0	3.0	Cls 0
2.0	0.0	Cls 1

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 class of the dataset. 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 fault diagnosis in Wireless Sensor Network's data. 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.

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 Int. 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 Int. Conf. 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, S., Jing, W., Fault Diagnosis of Aeroengine Sensor Based on Support Vector Machine, *2011 Third International Conference on Measuring Technology and Mechatronics Automation*, pp. 186-189, 6-7, January, 2011, Shanghai, China.

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.

Yuan, H., The Fault Diagnosis Research on Nonlinear Feature Extraction with Kernel Technology, *2011 Third International Conference on Measuring Technology and Mechatronics Automation*, pp. 811-814, 6-7, January, 2011, Shanghai, China.