Evolving class for SVM's incremental learning

Ahmed Benzerrouk, Brigitte Chebel Morello, Noureddine Zerhouni
Laboratoire d'automatique de Besançon, France.
25, rue Alain Savary.25 000 Besançon, France
ahmed.benzerrouk@lasmea.univ-bpclermont.fr
{bmorello, zerhouni}@ens2m.fr

Abstract

The good generalization performance of support vector machines (SVM) has made them a popular tool in artificial intelligence community. In this paper, we prove that SVM multi class algorithms are not equivalent for all classification problems. we present a new approach for incremental learning using SVM that create a rejection class which would be interesting for fault diagnosis where fault classes usually evolve with time: It is when some new samples may be rejected by all the current classes. Hence, these samples may correspond to a new fault (a new class) which may appear after the first training step.

1. Introduction

Most of artificial intelligence methods used in fault diagnosis have some disadvantages such as local optimal solution, low convergence rate, and poor generalization performance especially when database contains few samples. The lack of these one leads to wrong diagnosis or make it impossible.

Vapnik's work in statistical learning theory [1] leads to a new machine learning called Support Vector Machines. This one solves the problem of local optimal solution, convergence rate and has a good generalization performance even if the database has not a big number of samples. In the last case, incremental learning is used to improve the model when samples still arrive. In fault diagnosis, we notice that fault classes may change with the age of a machine. Hence, the current model will not be able to classify some new samples because they do not belong to any of the current classes already learned. In this paper, we propose to create a rejection class in evolving problems for samples that they have not been classified in any class of the current classifier.

The paper is organized as follows: the next section, section 2, reviews the SVM theory as its original form which is a binary classifier: we take the big lines from section 2 of [2]. In section 3, we see how SVM were generalized to multi class problems. We give a state of art on SVM incremental learning in section 4. Our idea is introduced in section 5 and we conclude in section 6.

2. Support Vector Machine (according to [2])

The idea of the pattern recognition with SVM is to project sample space into a high-dimensional eigenspace to find optimal separating hyper-planes of the original sample set without increasing calculation complexity. In this section we briefly describe the SVM algorithm and its motivation. A more detailed description of SVM can be found in [1].

We start from a simple case of two classes that is linearly separable.

Assuming that we have a data set:

$$D = \{(x_i, y_i)\} \quad (i = 1\dots l), \quad y_i \in \{-1, 1\} \qquad (1)$$

Where x_i is input sample and y_i is output class, y_i has two values (-1 or +1) that stand for two classes, l is the sample number. We wish to determine, among all linear separating planes that separate the input samples into two classes, which separating plane will have the smallest generalisation error. As shown in Figure 1, rings and diamonds stand for these two classes of sample points respectively; H is a separating plane. H1 and H2 are the planes that are parallel to H and respectively pass through the sample points closest to H in these two classes, the distance between H1 and H2 is defined as margin. The optimal separating plane that has the smallest generalisation error is the one that not only correctly separates all sample points into these two classes but also leaves the largest margin between H1 and H2.

If the two classes are not perfectly linearly separable we can still look for an optimal separating plane that maximizes the margin between H1 and H2 and minimizes the classifying error. The optimal separating plane is a linear classifier that has the classifying function as follows:

$$f(x) = \sum_{i=1}^{l} \alpha_i . y_i . x^T . x_i + b \qquad (i=1, \ldots, l)$$

H has as equation: f(x) =0 where as H1 and H2 have as equations *f(x)* =1 and *f(x)* = -1 respectively.

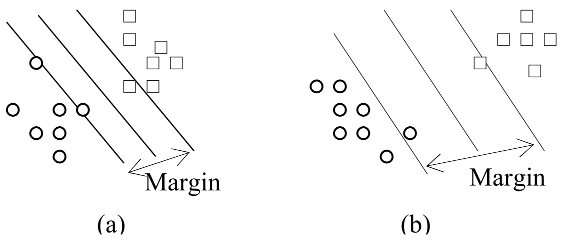


Figure 1. A separating plane with small margin (a) and with a larger margin (b)

To know the class of a sample x_i, we calculate $f(x_i)$, the Lagrange coefficient α_i is the solution of the following quadratic programming (QP) problem:

$$\begin{cases} \max\limits_{\alpha \in R^l} \ \frac{1}{2}\alpha^T G \alpha + \mathbf{1}^{T} \alpha \\ \text{Subject to} \quad y^T . \alpha = 0. \\ \qquad 0 \leq \alpha_i \leq C \end{cases}$$

Where $(\alpha) = \alpha_i$, $1_i = 1$, $G_{ij} = y_i y_j x_i^T x_j$

C is a penalty constant for those sample points misclassified by the optimal separating plane. Its role is to strike a proper balance between the calculation complexity and the classifying error. When $C \rightarrow +\infty$, it is the ideal case in which all samples are theoretically correctly separated by the optimal separating plane and there is no classifying error at all, but the calculation complexity is the biggest. It is found that only a few coefficients α_i are not zero, and since every coefficient α_i corresponds to a particular sample point, this means that only the sample points whose corresponding α_i are not zero determine the optimal separating plane. Only these few sample points, that are called support vectors, affect the classification result; while other sample points could be removed from the sample set and the optimal separating plane would be almost unaltered. These support vectors are usually few in the sample set.

Since practical problems are generally not linearly separable, SVM has to be developed for the classification of nonlinear problems. By projecting the original sample space into a high-dimensional eigenspace with a kernel function $K(x,x_i)$, the nonlinear separable problem becomes linearly separable in the eigenspace. The SVM classifier in the eigenspace has the classifying function as follows:

$$f(x) = \sum_{i=1}^{l} \alpha_i . y_i . k(x,x_i) + b \quad (i=1, \ldots, l),$$

In Table 1, some kernel functions $K(x,x_i)$ proposed by Vapnik are listed. Polynomial and Gaussian are the most used one.

Table 1. Some possible kernel functions and the type of classifiers they define.

Type of classifier	Kernel functions
Polynomial of degree d	$K(x,x_i)= (x^T.x + n)^d$
Gaussian	$K(x,x_i)= \exp(-\|x-x_i\|^2/2\sigma^2$
Multi-layer perceptron	$K(x,x_i)= \tanh (x^T.x + \theta)$

3. The SVM multi class algorithms

3.1 State of art

3.1.1 The one-to-one algorithm [7]. This algorithm proposes to make a binary classifier between every two classes. It is to train every class with all the other classes one by one. With n classes, n.(n-1)/2 binary classifiers are made (Figure 2.a).

3.1.2 The one-to-rest algorithm [8]. This algorithm proposes to make a binary classifier with each class against all the other classes. With n classes, n binary classifiers are made (Figure 2.b).

The disadvantage of these algorithms is the ambiguous regions where samples in "?" are classified. These samples have not been classified in any of the 3 possible classes.

There are some solutions to compensate these ambiguous regions like the vote where the sample is classified according to the number of times it occurred in each class (one-to-one) or the distance where the sample is classified in the closest class (one-to-rest).

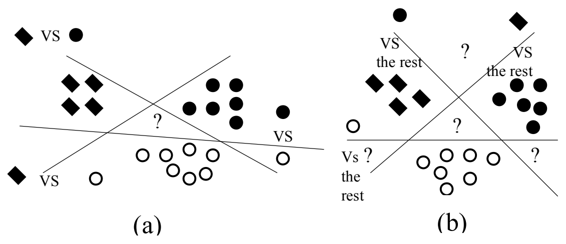


Figure 2. Classification with one to one algorithm (a) and one to rest algorithm (b)

3.1.3 The one to others algorithm [2]. Yuan and al, proposed an algorithm where there is no ambiguous regions. This algorithm is described as follow: take a top-priority class from the k classes of the problem as a category, take the rest (k - 1) classes as another category, construct a two-class SVM classifier; Next, this top-priority class is excluded, and then we have a case of (k-1) classes, take a top-priority class from those (k-1) classes as a category, and take the rest (k-1)-1 classes as another category, and construct a second two-class SVM classifier, and so on till the last two-class SVM classifier is constructed (Figure. 3).

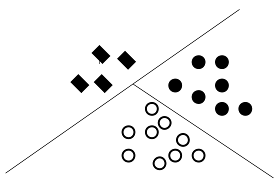


Figure 3. One-to-others algorithm for a 3 class problem

3.2 Comparison

Many toolboxes were developed for SVM such LibSVM by Chang and Lin (2003) and simpleSVM developed by Loosli (2004) on MATLAB that we use to compare results of different algorithms. However, one-to-others algorithm does not exist on this toolbox: we had to implement it distinguishing the two following cases:

1. One-to-others algorithm respecting points' distribution so that the most frequent class is categorized at first (however, in fault diagnosis the most critical class can be considered as a top-priority [2]).
2. One-to-others algorithm without respecting points' distribution.

Results are summarized in Table 2.

Table 2. Comparison of different multi class SVM algorithms

Data Base	SVM multi class algorithm	SVM prameters	Recognition rate
Balance-Scale: 3 class problem, 242 training samples of class 2, 134 training samples of class 3, 39 training samples of class 1, 208 testing samples.	One-to-one.	C=1 Kernel function : polynomial of degree 2.	90.09%
	One-to-rest.	C=1 Kernel function : polynomial of degree 2.	93.75%
	One-to-others without respecting points' distribution. Order of categorization: 1-2-3.	C=1 Kernel function : polynomial of degree 6.	94.71%
	One-to-others respecting points' distribution. Order of categorization: 2-3-1.	C=1, Kernel function :poly nomial of degree 3.	97%
Iris plant: 3 class problem, 35 training samples of class 1, 35 training samples of class 2, 30 training samples of class 3, 50 testing samples.	One-to-one.	C=1 Kernel function : polynomial of degree 2.5.	98%
	One-to-rest.	C=1 Kernel function : polynomial of degree 2.	96%
	One-to-others.	C=1, Kernel function :poly nomial of degree 2.	100%

It can be observed that one-to-others algorithm categorizing the most frequent class (of the training set) at first, gives better results than one-to-others algorithm generating randomly hyper planes between classes (see balance-scale database's result). Iris plant database has a homogenous distribution on classes and the class order in categorizing does not affect the model's performance.

Generally, all these algorithms give satisfactory results, but according to the problem, the algorithm's choice is very important to have the optimal and the best model.

1. In none evolving class problems, one to others algorithm has no ambiguous zone and gives the best recognition rate. We note, however, that when classes are hierarchic (in fault diagnosis, the most dangerous fault is the superior class [2]) it's advisable to start by

categorizing the top-priority one. Otherwise, frequency of classes in the training set is considered as the hierarchical criterion as seen in balance scale database.
2. In evolving class problems, one-to-rest algorithm seems more suitable thanks to its common ambiguous zone. We see in section 4 and 5 how to use this feature as an advantage.

4. Incremental learning algorithm

Incremental SVM learning is frequently used to reduce the storage cost by discarding useless samples (generally samples which are not support vectors) especially for huge databases, but also to speed up successive learning by using historical learning results.

Syed and al [3] used an incremental learning by partitioning the training datasets into subsets. At each incremental step, only support vectors are added to the next subset for the next training step. He showed that this incremental learning gives the same results as a full training.

Mitra and al [4] developed an incremental learning for SVM motivated from the condensed nearest neighbour classification technique: it is to condensate a data by choosing randomly a small number of samples to train SVM. Both a percentage of classified and misclassified data (taken from the remaining set) are retained with previous support vectors for the next training.

Exceeding margin technique given in [5] trains SVM and when new data $\{(x_i, y_i)\}$ arrives it is loaded in memory. The algorithm checks if (x_i, y_i) exceeds the margin of the previous SVM, ie, if $y_i.f(x_i) \leq 1$. If the condition is satisfied, the point is kept, otherwise it is discarded. When a given number n of points exceeding the margin is collected: SVM is updated using the support vectors of the current model and the n points.

Wenhua and Jian developed an incremental SVM learning based on **K**arush **K**uhn and **T**ucker conditions:(KKT) when a new set (incremental set) is available and should be used to update the current model, a second SVM is trained by the incremental set. Then, samples in the old set which violate the KKT conditions (Table 3) of the new SVM classifier and samples in the incremental set and violate the KKT conditions of the old classifier form the new training set of the updated model of SVM. More information on the use of KKT conditions in this algorithm is available in [6].

Table 3. Karush Kuhn and Tucker conditions

$y_i.f(x_i)>1$	$\alpha_i= 0$
$y_i.f(x_i)<1$	$\alpha_i=C$
$y_i.f(x_i)=1$	$0<\alpha_i<C$

Diehl and Cauwenberghs [7-8] used KKT conditions to increment a new example *l*. In [7], Cauwenberghs increments this point by increasing α_l *until* one of the KKT conditions relative to this point is violated or old points KKT conditions are perturbed by the incremented point. Diehl [8] increments the unlearned example preserving the KKT conditions of previous data. The KKT conditions are maintained by varying the support vector coefficients in response to the perturbation imparted by the incremented new coefficients.

In fault diagnosis, the training set may be initially poor. Hence, SVM is trained with this dataset and incrementally updated when new samples are available. These samples will be classified by the current model. However, we propose **to keep** new points classified in the common ambiguous zone **in a rejection class** instead of reviewing immediately our model (section 5).

5. Our proposal: the rejection class algorithm

5.1 Rejection class algorithm

When new samples are rejected by all the current classes, they are classified in the common ambiguous zone that we propose to consider as a rejection class. These samples may correspond to a new fault (a new class) which may appear with time because they do not really belong to any class of the current classifier. If these samples' number reach a threshold, the current model need to be updated according to the evolution of the machine faults because the old model is not valid any more since a possible new class is not considered in its previous training step.

We resume in figure 4 the reject zones of one-to-rest algorithm numbered from 1 to 4. It can easily be observed that future points classified in zones 1, 2, and 3 are not completely rejected. For example, points in zone 1, are one time classified as diamonds by $f_1(x)$ and one time as black rings by $f_2(x)$. So, it can be classified as a diamond or a ring according to the closest class. The same reasoning can be applied to zone 2 and 3.

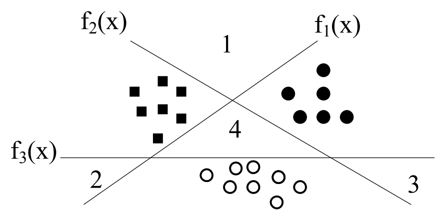
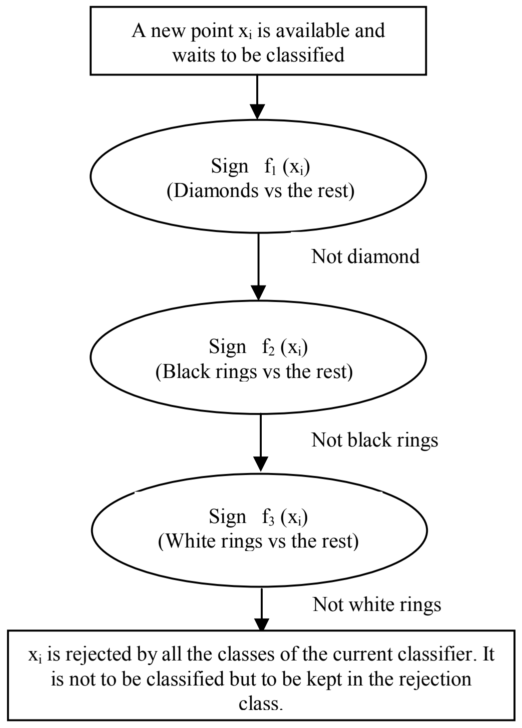


Figure 4. The one-to-rest algorithm with numbered reject zones

However, points in zone 4 are not classified in any class of the current model. Figure 5 shows how a point x_i can be rejected by all the classes and will be classified in zone 4.



A new point x_i is available and waits to be classified
Sign $f_1(x_i)$ (Diamonds vs the rest)
Not diamond
Sign $f_2(x_i)$ (Black rings vs the rest)
Not black rings
Sign $f_3(x_i)$ (White rings vs the rest)
Not white rings
x_i is rejected by all the classes of the current classifier. It is not to be classified but to be kept in the rejection class.

Figure 5. A rejected point by the current classifier

5.2 Experimental results

We work on wine recognition and iris plant databases. In each data, we train a one-to-rest SVM classifier. After that, we create new samples and classify them by the current classifier. We keep those who were classified in the common ambiguous zone as points of a rejection class.

Results are summarized in Table 4.

Table 4. Rejection class results

Data Base	SVM parameters	Recognition rate	Rejected points
Wine recognition: 3 class problem, 119 training samples, 59 testing samples.	C= 1, Kernel function: polynomial of degree 2.	96.67%	The 5 new points.
Iris plant: 3 class problem, 100 training samples, 50 testing samples.	C = 1 Kernel function: Polynomial of degree 2.	100 %	4 of the 8 new points.

6. Conclusion

We have seen how to use the properties of an SVM binary classifier for multi class problems. In non evolving classes, we checked that one-to-others algorithm gives better results since there are no reject zones and points are classified in their suitable classes. However, to enhance its result, we have to start by categorizing classes according to the hierarchical criterion (dangerous fault, frequent class…).

But, evolving class problems, like fault diagnosis, drive us to think for a SVM classifier able to predict even the evolution of the classes: the one-to-rest algorithm has a common ambiguous zone that we consider as a rejection class which allow us to control the classes' evolution and help expert to predict new classes. When new points came in droves in this rejection class, we can warn the expert that a possible new class has been created. We avoid by this way updating unnecessarily and maybe wrongly the model: this can be a huge waste of time.

We have seen that our algorithm makes possible not only fault diagnosis but also new class prediction so that the expert can be warned early, before that he realizes it himself taking more time. Hence, we minimize the risk that the machine breaks completely down due to this unknown new fault.

If the expert confirms that a new class has been created, a new SVM should be training taking the new class into account.

7. References

[1] V.N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, USA, 1999.

[2] Sheng-Fa Yuan, Fu-Lei Chu, “Support vector machines-based fault diagnosis for turbo-pump rotor”, *Mechanical Systems and Signal Processing*, May 2006, pp. 939-952.

[3] N. Syed, H. Liu, and K.Sung, “Incremental learning with support vector machines”, *Proceedings of the workshop on Support Vector Machines at the International Joint Conference on Artificial Intelligence*, IJCAI-99, Stockholm, Sweden, 1999, pp. 352-356.

[4] P. Mitra, C. A. Murthy, S. K. Pal, “Data Condensation in Large Databases by Incremental Learning with Support Vector Machines”, *International Conference on Pattern Recognition (ICPR2000)*, Barcelona, Spain, 2000, pp. 712-715.

[5] C. Domeniconi, D. Gunopulos. “Incremental Support Vector Machine construction”, *Proceedings IEEE International Conference on data mining,* San Jose, USA, 2001, pp. 589 - 592.

[6] B. Dubuisson, *Diagnostic et reconnaissance des formes*, Hermès, Paris 1990.

[7] U.Krebel, “Pairwise classification and support vector machines”, *Kernel Methods – Support Vector Learning*, Cambridge, MA, 1999. pp. 255-268.

[8] V.N. Vapnik, *The Nature of Statistical Learning Theory*, Springer Verlag, New York, 1995.