

A fast multi-class SVM learning method for huge databases

Djeffal Abdelhamid¹, Babahenini Mohamed Chaouki² and Taleb-Ahmed Abdelmalik³

^{1,2} Computer science department, LESIA Laboratory,
Biskra University, Algeria

³ LAMIH Laboratory FRE CNRS 3304 UVHC, Valenciennes university, France

Abstract

In this paper, we propose a new learning method for multi-class support vector machines based on single class SVM learning method. Unlike the methods 1vs1 and 1vsR, used in the literature and mainly based on binary SVM method, our method learns a classifier for each class from only its samples and then uses these classifiers to obtain a multiclass decision model. To enhance the accuracy of our method, we build from the obtained hyperplanes new hyperplanes, similar to those of the 1vsR method, for use in classification. Our method represents a considerable improvement in the speed of training and classification as well the decision model size while maintaining the same accuracy as other methods.

Keywords: Support vector machine, Multiclass SVM, One-class SVM, 1vs1, 1vsR.

1. Introduction and related work

The support vector machine (SVM) method is, in its origin, binary. It is based on the principle of separation between the samples of two classes, one positive and one negative. In this form, the SVM method is very successful in several areas of application given the precision it offers. In practice, we find more applications with multi-class problems, hence the need to extend the binary model to meet multi-class problems. Existing methods currently try mainly to optimize two phases: a training phase and a classification phase. The first phase constructs the hyperplane, and the second uses it.

The evaluation of the methods is based on the evaluation of the performances of the two phases. Among the well known methods, there are methods for direct solution without using the binary SVM model as Weston & Watkins model [1], but which suffers, always, from some slowness and weak accuracy. The widely used methods are based essentially on the extension of the binary model, namely, the one-against-rest (1vsR) and the one-against-one (1vs1) methods. The 1vsR method learns for each class a hyperplane that separates it from all other classes, considering this class as positive class and all other classes

as negative class, and then assigns a new sample, in the classification phase, to the class for which it maximizes the depth. The 1vs1 method learns for each pair of classes a separating hyperplane, and uses the voting lists or decision graphs (DAG) to assign a new sample to a class [2,3].

In [4,5,6,7] comparative studies are conducted to assess the performances of these methods. According to the authors, the 1vs1 method is faster while the 1vsR method is more accurate. \\

In this paper, instead of the binary SVM, we propose to use the one-class SVM (OC-SVM) that provides a hyperplane for each class that separates it from the rest of the space. For each class, we learn a hyperplane from only its samples. Then in the classification phase, we build for each class a new two-class hyperplane which separates it from all other classes. This two-class hyperplane is calculated from the previous one-class hyperplane and the closest sample of the other classes. The new two-class hyperplane is a shift of the one-class hyperplane, it is situated between the farthest misclassified sample, of the target class, from the hyperplane, and nearest sample, belonging to other classes, to the hyperplane. Our technique speeds up the training time and classification time, and reduces the decision model size compared to classic methods, while keeping very close accuracy. Our results were validated on toys and then on databases of UCI site [8]. The rest of the paper is organized as follows: section 2 introduces the binary SVM and Section 3 introduces the multi-class methods based on the binary model, namely, 1vsR and 1vs1. In section 4, we present the OC-SVM model and then we present our method in Section 5. The results and their discussion are presented in Section 6, and a conclusion in Section 7.

2. Binary SVM

The binary SVM solves the problem of separation of two classes, represented by n samples of m attributes each

[9,10]. Consider the problem of separating two classes represented by n samples:

$$\{(x_1, y_1), \dots, (x_n, y_n)\}, x_i \in \mathbb{R}^m, y_i \in \{-1, +1\}$$

Where x_i are learning samples and y_i their respective classes. The objective of the SVM method is to find a linear function f (equation 1), called *hyperplane* that can separate the two classes:

$$\begin{cases} f(x) = (x \bullet w) + b; \\ f(x) > 0 & \Rightarrow x \in \text{class} + 1 \\ f(x) < 0 & \Rightarrow x \in \text{class} - 1 \end{cases} \quad (1)$$

Where x is a sample to classify, w is a vector and b is a bias.

We must therefore find the widest margin between two classes, which means minimizing w^2 . In cases where training data are not linearly separable, we allow errors ξ_i (called slack variables) of samples from boundaries of the separation margin with a penalization parameter C and the problem becomes a convex quadratic programming problem:

$$\begin{cases} \text{Minimize} & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \\ \text{under constraints} & y_i(w^T x_i + b) \geq 1 - \xi_i; i = 1..n \\ & \xi_i \geq 0 \end{cases} \quad (2)$$

The problem of equation 2 can be solved by introducing Lagrange multipliers α_i in the following dual problem:

$$\begin{cases} \text{Minimize} & \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(x_i x_j) - \sum_{i=1}^n \alpha_i \\ \text{under constraints} & \sum_{i=1}^n \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \end{cases} \quad (3)$$

Hence, we can have the following decision function (hyperplane):

$$H(x) = \sum_{i=1}^n \alpha_i y_i K(x_i, x) + b \quad (4)$$

The function K is called *Kernel*, it is a symmetric function that satisfies Mercer conditions [4]. It can represent a transformation of the original input space in which data could be non-linearly separable to a new larger space where a linear separator exists. Solving the problem of equation 3 requires an optimization, especially when the number of samples is high. Among the optimization methods most commonly used, there is the SMO (Sequential Minimal Optimization) where the problem is broken into several sub-problems, each optimizes two α_i [11].

3. Multi-class SVM

Several techniques have been developed to extend binary SVM method to problems with multiple classes. Each of

these techniques makes a generalization of the abilities of the binary method to a multi-class field [2,3]. Among the best known methods, we can cite 1vsR and 1vs1.

3.1 One-against-rest (1vsR)

For each class k we determine a hyperplane $H_k(w_k, b_k)$ separating it from all other classes, considering this class as positive class (+1) and other classes as negative class (-1), which results, for a problem of K classes, to K binary SVMs. A hyperplane H_k is defined by the following decision function:

$$f_k(x) = \langle w_k \bullet x \rangle + b_k \quad (5)$$

This function allows to discriminate between the samples of the class k and the set of all other classes.

Classification: A new sample x is assigned to the class k^* that maximizes the depth of this sample. This class is determined by the decision rule of the equation 6.

$$k^* = \text{Arg}_{(1 \leq k \leq K)} \text{Max} f_k(x) \quad (6)$$

Figure 1 shows an example of separation of three classes.

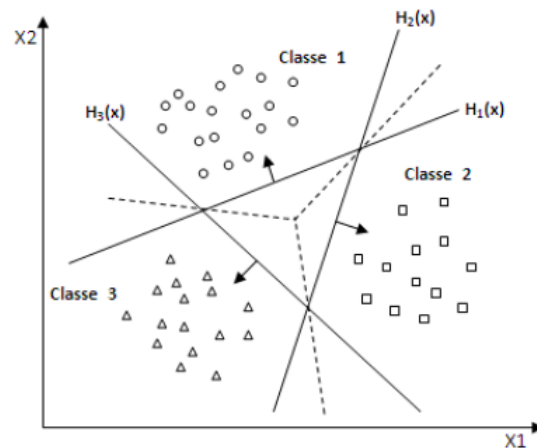


Fig. 1 One-against-rest approach

Interpreted geometrically, a new sample x is assigned to the class that corresponds to the farthest hyperplane. Thus, the space is divided into K convex regions, each corresponding to a class.

3.2 One-against-one (1vs1)

Training: Instead of learning K decision functions, the 1vs1 method discriminates each class from every other class. Thus $K(K-1)/2$ decision functions are learned. The

one-against-one approach is, thus, to build a classifier for each pair of classes (k, s) .

Classification: For each pair of classes (k, s) , the 1vs1 method defines a binary decision function $H_{ks}: x \in \mathcal{X}^m \rightarrow \{-1, +1\}$. The assignment of a new sample can be done by two methods; voting list or decision graph.

a. Voting list: We test a new sample by calculating its decision function for each hyperplane. For each test, we vote for the class to which the sample belongs. We define, for this, the binary decision function $H_{ks}(x)$ of equation 7.

$$H_{ks}(x) = \text{sign}(f_{ks}(x)) = \begin{cases} +1 & \text{if } f_{ks}(x) > 0; \\ 0 & \text{else} \end{cases} \quad (7)$$

Based on the $K(K-1)/2$ binary decision functions, we define other K decision functions (equation 8):

$$H_k(x) = \sum_{s=1}^m H_{ks}(x) \quad (8)$$

The classification rule of a new sample x is given, then, by the equation 9:

$$k^* = \text{Arg}_{(1 \leq k \leq K)} \text{Max} H_k(x) \quad (9)$$

Figure 2 is an example of classification of three classes.

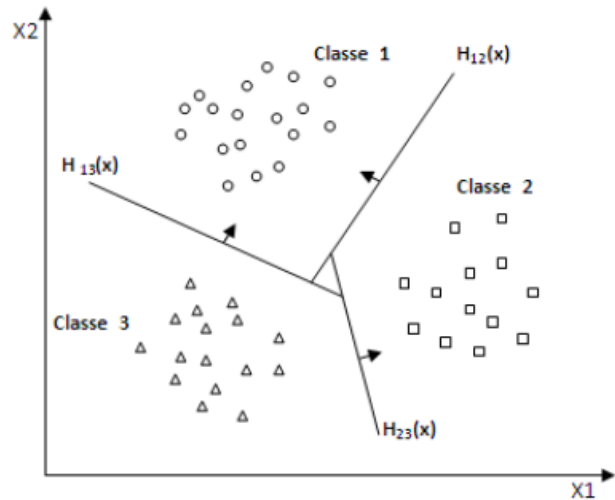


Fig. 2 One-against-one approach

b. Decision graphs: In this method, we define a measure E_{ks} of the generalization ability of the different obtained hyperplanes i.e for each pair of classes. This measure (equation 10) represents the ratio between the number of support vectors of the hyperplane and the number of samples of both classes.

$$E_{ks} = \frac{N_{vs}}{N_{examples}} \quad (10)$$

Before deciding about new samples, we construct a decision graph. We start with a list L containing all the classes, then take the two classes k and s which E_{ks} is maximum, and we create a graph node labeled (k, s) . We then create, in the same way, a left son of this node from

the list $L - \{k\}$ and a right son from the list $L - \{s\}$, and continue until the list L contains only one class. This gives the decision graph of figure 3, whose leaves are the classes and the internal nodes are the hyperplanes:

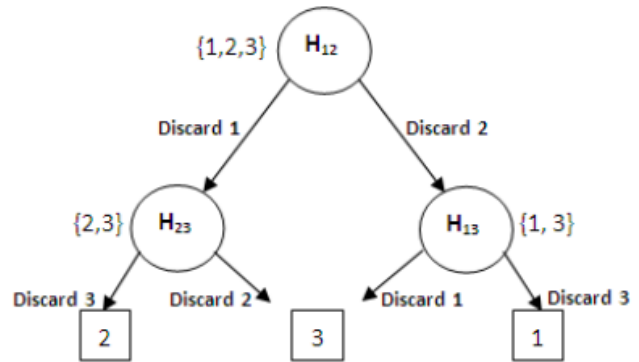


Fig. 3 Directed acyclic decision graph for three classes

A new sample x is exposed first to the hyperplane of the root, if the decision is positive then we continue with the left son, otherwise with the right son, until a leaf is reached. The reached leaf represents the class of the sample x .

4. One-class SVM

In the one-class SVM classification, it is assumed that only samples of one class, the target class, are available. This means that only the samples of the target class can be used and no information on other classes is present. The objective of the OC-SVM is to find a boundary between the samples of the target class and the rest of space. The task is to define a boundary around the target class, so that it accepts as many target samples as possible [12]. The one-class SVM classifier considers the origin as the only instance of negative class, and then tries to find a separator between the origin and samples of the target class while maximizing the margin between the two (cf. figure 4).

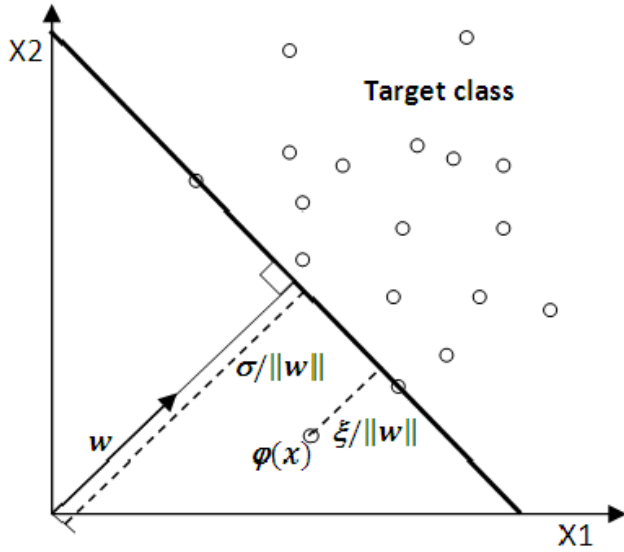


Fig. 4 One-class SVM with maximum margin

The problem is to find the hyperplane that separates the target samples of the origin while maximizes the distance between them. The problem is modeled by the primal quadratic programming problem of equation 11.

$$\begin{cases} \min_{w, \xi, \rho} \frac{1}{2} \|w\|^2 + \frac{1}{vN} \sum_{i=1}^l \xi_i - \rho \\ \langle w, \phi(x_i) \rangle \geq \rho - \xi_i \\ \xi_i \geq 0 \quad i = 1, 2, \dots, l \end{cases} \quad (11)$$

Where N is the number of samples of the target class, (w, ρ) parameters to locate the hyperplane, ζ_i the allowed errors on samples classification, penalized by the parameter v and ϕ is a transformation of space similar to the binary case. Once (w, ρ) determined, any new sample can be classified by the decision function of equation 12:

$$f(x) = \langle w, \phi(x_i) \rangle - \rho \quad (12)$$

x belongs to the target class if $f(x)$ is positive. In fact, solving the problem of the equation 11 is achieved by the introduction of Lagrange multipliers in the dual problem of equation 13:

$$\begin{cases} \text{Minimize}_{\alpha} & \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \\ \text{under constraints} & \sum_{i=1}^n \alpha_i = 1 \\ & 0 \leq \alpha_i \leq \frac{1}{v} \end{cases} \quad (13)$$

Where K is a kernel that represents the space transformation ϕ . Once the α_i are determined using an

optimization such as SMO [11], the decision function for any sample x is given by equation 14:

$$f(x) = \sum_{i=1}^l \alpha_i K(x_i, x) - \rho \quad (14)$$

Where ρ can be determined from a training sample x_i having $\alpha_i \neq 0$ by the equation 15:

$$\rho = \sum_j \alpha_j K(x_j, x_i) \quad (15)$$

5. Multi-class SVM method based on OC-SVM (OCBM-SVM)

5.1 Training

The method we propose in this paper, extends the OC-SVM to multi-class classification. We propose to learn for each class its own hyperplane separating it from the rest of the space. We learn so for K classes, K hyperplane, but unlike the 1vsR method, we use to find each hyperplane H_k only the samples of the class k which speeds up considerably the training time. Table 1 shows the cost of different methods in terms of the number of used hyperplanes, the number of samples used by each hyperplane, and the estimated time of training depending on the number of samples of a class N_c . We assume, for the sake of simplicity, that the classes have the same number of samples. The estimated time is based on the fact that each hyperplane H_k using N_k samples requires a time of βN_k^2 (β represents the conditions of implementation such as CPU speed, memory size,...etc.).

Table 1: Comparative table of the training times of the different methods

Method	# Hyperplanes	# samples/Hyperplane	Estimated time
1vsR	K	KN_c	$K^3 \beta N_c^2$
1vs1	$K(K-1)/2$	$2N_c$	$2\beta K^2 N_c^2$
OCBM-SVM	K	N_c	$K \beta N_c^2$

- The 1vsR method uses to determine each of the K hyperplanes all KN_c training samples, which leads to a training time of $K^3 \beta N_c^2$, where K represents the number of classes and β a constant related to the conditions of implementation, based on the fact that the SMO algorithm [11] used here is of complexity $O(N^2)$.
- The 1vs1 method calculates a hyperplane for each pair of classes, i.e $K(K-1)/2$ hyperplanes, and to determine a hyperplane separating two classes, we use the samples of these two classes $2N_c$, resulting in a training time of $2\beta K^2 N_c^2$.

- Our method requires the calculation of K hyperplanes, each separates a class from the rest of space. To determine each hyperplane, we use only the samples of one class, resulting therefore in a total training time of about $2KN_c^2$.

It is clear that:

$$K\beta N_i^2 < 2\beta K^2 N_i^2 < K^3 \beta N_i^2 \quad (16)$$

This means that the proposed method optimizes the training time compared to methods 1VsR and 1vs1.

5.1 Classification

OC-SVM method allows to find a hyperplane separating one class from the rest of space, this hyperplane allows to decide on membership of a new sample to this class. If the sample is above the hyperplane then it belongs to the class, but if it is below, we have no information on to what other class the sample belongs. To correct this situation, we propose to modify the obtained hyperplane to enhance the decision information in the case where the sample does not belong to the class. We propose to find for each hyperplane of a class, the closest sample among the samples of all other classes, then shift the hyperplane by a distance equal to half the distance between this sample and the misclassified sample, of the target class, that minimizes the decision function (the farthest one from the hyperplane) (cf. Figure 5).

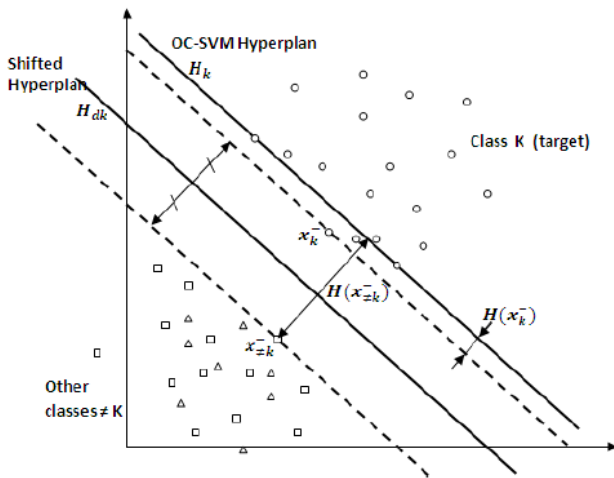


Fig. 5 Classification in OCBM-SVM method

More formally, let $H_k(x)$ be the decision function using the hyperplane of the k^{th} class. And let x_k^- be the misclassified sample of the class k farthest from the hyperplane H , and $x_{\neq k}^-$ the closest sample to the hyperplane H , belonging to a class different to k . The distance of x_k^- from the hyperplane is given by $H_k(x_k^-)$ and the distance between

the hyperplane and $x_{\neq k}^-$ is given by $H_{\neq k}(x_{\neq k}^-)$. The proposed shift is $[H_k(x_k^-) + H_{\neq k}(x_{\neq k}^-)]/2$, and the new decision function for a sample x can be calculated by the shifted hyperplane H_{dk} of the equation 17:

$$H_{dk}(x) = H_k(x) - \frac{(H_k(x_k^-) + H_{\neq k}(x_{\neq k}^-))}{2} \quad (17)$$

After calculating all the K shifted hyperplanes, the decision about the class k^* of a new sample x can be given by the discrete decision rule of equation 18:

$$k^* = \text{Arg}_{(1 \leq k \leq K)} \max H_{dk}(x) \quad (18)$$

Table 2 shows a comparison between the time of classification of a new sample by different methods. This time is calculated based on the number of support vectors of each hyperplane, which is equal, to the maximum, the total number of samples used to find the hyperplane.

Table 2: Comparative table of classification time of the different methods

Method	# Used hyperplanes	Estimated time
1vsR	K	$K^2 \beta N_c$
1vs1	$K(K-1)/2$	$K(K-1) \beta N_c$
DAG	$(K-1)$	$2(K-1) \beta N_c$
OCBM-SVM	K	$K \beta N_c$

- The 1vsR method uses K hyperplanes, and to test a sample, it is evaluated for all hyperplanes. Knowing that each hyperplane contains a number of support vectors equal to the maximum total number of samples, the estimated time to find the class of a sample is $K^2 \beta N_c$, where β is a constant that represents the conditions of implementation.
- For the method 1vs1 using the voting list, the classification estimated time is $K(K-1) \beta N_c$.
- For the method 1vs1 using decision graphs, the classification estimated time is $2(K-1) \beta N_c$.
- The estimated time for our method is $K \beta N_c$, because it tests the sample for K hyperplanes containing, at maximum, KN_c support vectors.

If we eliminate from the column of estimated time in Table 2 the factor βN_c , we note that the method OCBM-SVM optimizes the classification time. But this depends always on the number of support vectors obtained by the training method which can be very important compared to the number of classes K . Higher the number of training samples and the number of classes, the greater the advantage of the OCBM-SVM method, since it uses the minimum of samples, making it suitable for large databases.

5.3 Model size

The method we propose in this paper, extends the OC-SVM to multi-class classification. We propose to learn for each class its own hyperplane separating it from the rest of the space. We learn so for K classes, K hyperplane, but unlike the 1vsR method, we use to find each hyperplane H_k only the samples of the class k which speeds up considerably the training time. Table 1 shows the cost of different methods in terms of the number of used hyperplanes, the number of samples used by each hyperplane, and the estimated time of training depending on the number of samples of a class N_c . We assume, for the sake of simplicity, that the classes have the same number of samples. The estimated time is based on the fact that each hyperplane H_k using N_k samples requires a time of βH_k^2 (β represents the conditions of implementation such as CPU speed, memory size,...etc.).

6. Experiments

6.1 Used data

Our method was first tested on examples of Toy type (2 dimensions) that we have chosen of different complexities. Then we have tested it on benchmarks of multi-class classification databases from the site "Machine Learning Repository UCI" [8]. The used databases are shown in the following Table 3:

Table 3: Databases used for testing

Base	Domain	N_{att}	N_{class}	N_{train}	N_{test}
PageBlocks	Web	10	5	389	5090
Segment	Image processing	19	7	500	1810
Vehicle	Industry	18	4	471	377
Abalone	Industry	8	27	3133	1044
Letter	Character recognition	16	26	2000	10781
OptDigits	Industry	64	10	3824	1797

N_{att} is the number of attributes, N_{class} is the number of classes in the database, N_{train} is the number of samples used for training and N_{test} is the number of samples used for testing.

6.2 Materials and evaluation criteria

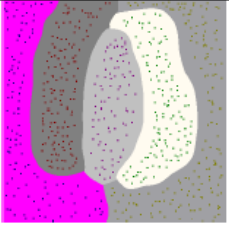
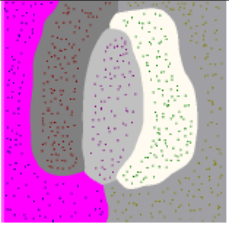
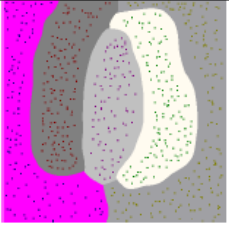
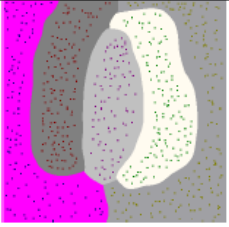
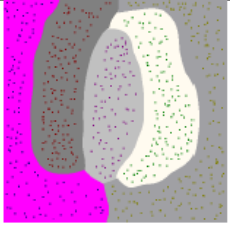
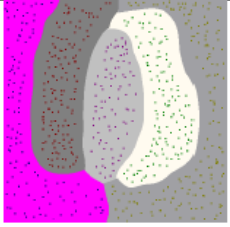
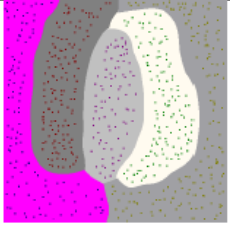
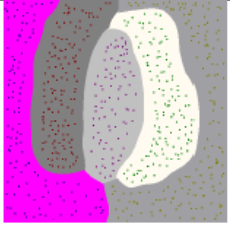
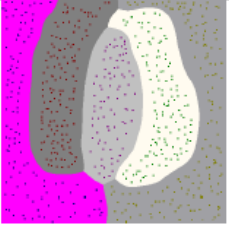
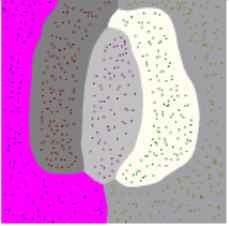
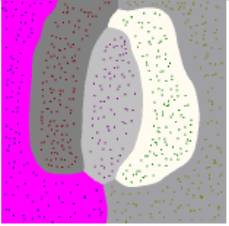
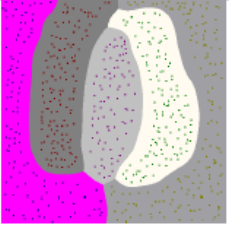
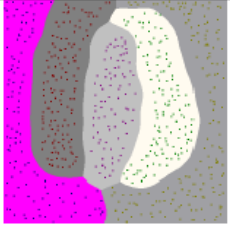
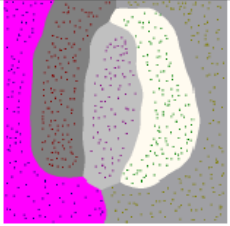
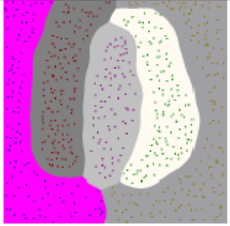
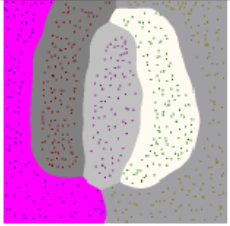
The proposed method was tested on a Dual-core de 1.6 GHZ machine with 1GB of memory. The used kernel is RBF, and the optimization method is the SMO algorithm [11]. The evaluation criteria of the performances of our method are the training time in seconds $T_r(s)$, the classification time in seconds $T_c(s)$,

the recognition rate R and the size of the obtained model.

6.3 Results

The first tests have been performed on toys of different complexities and have shown the advantages of OCBM-SVM method compared to other methods. In the example of table 4, we took 642 samples belonging to five classes. The results show that our OCBM-SVM method reduced training time to 0.109 second without losing accuracy, while the 1vs1 and 1vsR methods have given respectively 0.609 and 28.25 seconds. Even, the size of the resulting model was reduced to 18.290 KB. The classification time was 22.625 which is close to that of the method of decision graphs (DAG) with the remark that the number of samples and the number of classes are small.

Table 4: Results on a toy

1vs1				DAG			
							
Tr(s)	Tc (s)	R(%)	Size(KB)	Tr(s)	Tc (s)	R(%)	Size(KB)
8.609	41.922	100	24.71	8.609	20.610	100	24.71
1vsR				OCBM-SVM			
							
Tr(s)	Tc(s)	R(%)	Size(KB)	Tr(s)	Tc(s)	R(%)	Size(KB)
28.250	44.890	100	23.714	0.109	22.625	100	14.282

The tests performed on the databases of the UCI site also confirm the theoretical results. Table 5 summarizes the results obtained on testing databases presented in the table 3.

Indeed, in all tested databases, OCBM-SVM method greatly improves the training time and model size, especially in large databases. For the *Abalone* database, our OCBM-SVM method reduced the training time from 3954.047 seconds to 1.5 seconds and the size of the model from 2996,002 KB to 226,954 KB. In the case of *OptDigits* database, training time was reduced from 16981.984 seconds for 1vs1 method and 68501.407 seconds (over 19 hours) for 1vsR method, to

only 126,593 seconds while maintaining accuracy better than the method 1vs1. For the same database, the size of the model was also reduced from 3749.174 KB for the 1vs1 method and 2148,554 KB for 1vsR to 555,314 KB.

Table 5: Results on different databases

Base	Parameters	1vs1(Vote)	1vs1(DAG)	1vsR	OCBM-SVM
PageBlocks	<i>Tr(s)</i>	332	332	1105.516	8.531
	<i>Tc(s)</i>	6.922	4.718	7.906	8.109
	<i>R(%)</i>	93.33	93.33	93.31	93.31
	<i>Size(KB)</i>	135.726	135.726	168.986	34.170
	<i>#Hyperplanes</i>	10	10	5	5
Segment	<i>Tr(s)</i>	51.860	38.875	105.172	0.079
	<i>Tc(s)</i>	1.828	2.844	3.859	2.843
	<i>R(%)</i>	78.23	77.79	76.85	76.24
	<i>Size(KB)</i>	266.346	266.346	177.522	78.162
	<i>#Hyperplanes</i>	21	21	7	7
Vehicule	<i>Tr(s)</i>	481.313	481.313	1127.172	0.171
	<i>Tc(s)</i>	0.812	0.656	0.484	0.672
	<i>R(%)</i>	69.41	69.41	72.07	71.80
	<i>Size(KB)</i>	202010	202010	259546	71066
	<i>#Hyperplanes</i>	6	6	4	4
Abalone	<i>Tr(s)</i>	3954.047	3954.047	11324.652	1.5
	<i>Tc(s)</i>	14.125	6.421	9.322	5.282
	<i>R(%)</i>	21.64	21.16	24.89	25
	<i>Size(KB)</i>	2996.002	2996.002	5478.347	226.954
	<i>#Hyperplanes</i>	351	351	27	27
Letter	<i>Tr(s)</i>	4399.344	4399.344	34466.938	52.703
	<i>Tc(s)</i>	246.453	44.500	98.484	10.766
	<i>R(%)</i>	82.91	82.91	84.83	79.59
	<i>Size(KB)</i>	6102.174	6102.174	2713.442	214.034
	<i>#Hyperplanes</i>	325	325	26	26
OptDigits	<i>Tr(s)</i>	16981.984	16981.984	68501.407	126.593
	<i>Tc(s)</i>	54.022	14.500	32.15	24.578
	<i>R(%)</i>	93.16	93.16	97.16	93.56
	<i>Size(KB)</i>	3749.174	3749.174	2148.554	555.314
	<i>#Hyperplanes</i>	45	45	10	10

The proposed method keeps a classification time, in the case of large databases, close to that of the method DAG representing the fastest method in terms of classification time. Indeed, we note, in databases with a number of samples less than 1000 (case of *PageBlocks*, *Segment* and *Vehicule*) that the classification time obtained by some methods is better than ours. This is due to that in the databases of small size, the preparation work of classification structures (lists in DAG and shifts in OCBM-SVM) is important compared to the essential task of calculating the decision functions. In large databases, with number of samples greater than 1000 and number of classes greater than 10 (case of *Abalone*, *Letter*, *OptDigits*), the improvement of classification time is remarkable. Indeed, for the *Abalone* database, our method yielded a classification time of 5.282 seconds against 6.421 seconds for DAG method the best of the others.

Also for the *Letter* database, the classification time was 10,766 seconds against 44.5 seconds for DAG. In the *OptDigits* database, the obtained time by the DAG method was better than the OCBM-SVM method, this can be explained by the number of support vectors obtained by each training method which has an important influence on the classification time.

With all its advantages, our method preserves a recognition rate in the range of rates obtained by other methods and sometimes better (case of *Abalone* database).

7. Conclusion

In this paper, we presented a new method for multiclass learning with support vector machine method. Unlike classic methods such as 1vs1 and 1vsR extending the principle of binary SVM, our method (denoted OCBM-SV) extends the principle of one-class SVM method. And to achieve the generalization ability of binary SVM, we changed the hyperplanes obtained by the one-class method to take into account information of other classes. The obtained results show great improvement in training time and decision model size. Our method also allows improving the classification time (decision making) of new samples by reducing of the number of support vectors of the decision model. The obtained accuracy is very close to that of other methods and sometimes better.

8. References

- [1] U. Dogan, T. Glasmachers, C. Igel, "Fast Training of Multiclass Support Vector Machines", Technical Report no. 03/2011, Faculty of science, university of Copenhagen, 2011.
- [2] S. Abe, "Analysis of multiclass support vector machines", In Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA'2003), pp. 385-396, 2003.
- [3] Y. Guermeur, "Multi-class Support vector machine, Theory and Applications", HDR thesis, IAEM Lorraine, 2007.
- [4] Y. Liu, R. Wang, Y. Zeng, H. He, "An Improvement of One-against-all Method for Multiclass Support Vector Machine", 4th International Conference: Sciences of Electronic, Technologies of Information and telecommunications, March 25-29, 2007, TUNISIA.
- [5] N. Seo, "A Comparison of Multi-class Support Vector Machine Methods for Face Recognition", Research report, The University of Maryland, Dec 2007.
- [6] G. Anthony, H. Gregg, M. Tshildzi, "Image Classification Using SVMs: One-against-One Vs One-against-All", Proceedings of the 28th Asian Conference on Remote Sensing, 2007.
- [7] M. G. Foody, A. Mathur, "A Relative Evaluation of Multiclass Image Classification by Support Vector Machines", IEEE Transactions on Geoscience and Remote Sensing, V. 42 pp. 13351343, 2004.
- [8] A. Frank, A. Asuncion, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml], Irvine, CA: University of California, School of Information and Computer Science, 2010.