OVA Tree Multiclass Framework for Support Vector Machine

Boutkhil Sidaoui, Kaddour Sadouni

***Abstract*—In this paper we propose and examine the performance of a framework for solving multiclass problems with Support Vector [1]Machine (SVM). Our methods based on the principle binary tree, leading to much faster convergence and compare it with very popular methods proposals in the literature, both in terms of computational needs for the feedforward phase and of classification accuracy. The proposed paradigm builds a binary tree for multiclass SVM, using the technical of portioning by criteria of natural classification: Separation and Homogeneity, with the aim of obtaining optimal tree. The main result, however, is the mapping of the multiclass problem to a several bi-classes sub-problem, in order to easing the resolution of the real and complex problems. Our approach is more accurate in the construction of the tree. Further, in the test phase OVA Tree Multiclass, due to its Log complexity, it is much faster than other methods in problems that have big class number. In this context, two corpus are used to evaluate our framework; TIMIT datasets for vowels classification and MNIST for recognition of handwritten digits. A recognition rate of 57%, on the 20 vowels of TIMIT corpus and 97.73% on MNIST datasets for 10 digits, was achieved. These results are comparable with the state of the arts, in particular the results obtained by SVM with one-versus-one strategy. In addition, training time and number of support vectors, which determine the duration of the tests, are also reduced compared to other methods. However, these results are unacceptably large for the real application of speech and handwritten digits recognition task.**

***Keywords*—Machine Learning, SVM, Classification, Binary Tree, Separation, Homogeneity.**

I. Introduction

KERNEL Methods and particularly Support Vector Machines (SVMs) (SVM) introduced during the last decade in the context of statistical learning, received lately a lot of attention of the part of the community of research in algorithms of machine learning, for his solid theoretical foundations and their practical successes on concrete problems [1], [2], [3], [36, 37]. SVMs are arguably the single most important development in supervised classification of recent years. SVMs often achieve superior classification performance compared to other learning algorithms across most domains and tasks; they are fairly insensitive to the curse of dimensionality and are efficient enough to handle very large-scale classification in both sample and variables.

SVMs [2], [4], [5], have been successfully used for the solution of a large class of machine learning tasks [6], [7] such as categorization, prediction, novelty detection, ranking and clustering. The success of Support Vector Machines on a number of high dimensional tasks has prompted a renewed interest in kernel methods. The time it takes to classify a new example using a trained support vector machine is proportional to the number of support vectors. While SVMs are a very robust and powerful technique for supervised classification, the large size and slow query time of a trained SVM is one hindrance to their practical application especially for Multiclass problems.

Where SVM was originally developed for binary problems and its extension to multi-class problems is not straightforward. How to effectively extend it for solving multiclass classification problem is still an on-going research issue. Several multiclass methods have been proposed which successfully help to alleviate this problem. The most popular and widely successful methods for applying SVMs to multiclass classification problems usually decompose the multi-class problems into several two-class problems that can be addressed directly using several.

Many real–world applications consist of multiclass classification problems as: recognition of handwritten digits and automatic Speech Recognition (ASR). ASR has been the focus of both the machine learning and speech communities for the past few decades due to its importance in any conceivable man machine interface (MMI) [8]. ASR is considered as the most difficult and challenging problem to be solved for many years to come. The large variability and the richness of voice data represent a fertile field to evaluate the performance of recognition systems. Many approaches have been proposed towards the goal of improving the performance of ASR. Hidden Markov Models (HMM) are the most widely used recognizers for ASR, due to their ability of efficiently modeling the sequential nature of the speech frames [9], [10], [11]. Class of such methods [12], [13], [14], [15], [16], [17] focuses on new techniques for feature extraction. Others, keep the MFCC features along with derivative information, but use novel powerful discriminative methods [18], [19], [20], and [21]. Our contributions use the standard MFCC with powerful discriminative methods (SVM). In this paper, the task of vowels classification is used to evaluate our approach.

The remainder of the paper is organized as follows. We begin this paper with a brief introduction about multiclass problems and methods of machine learning. In section 2, kernel methods SVM will be discussed for problems classification. In the following section, we provide a brief state of art of the multi-class approaches, the most popular, and published in the literatures. In the next section, we

Manuscript received October 13, 2012; revised December 30, 2012.
Boutkhil SIDAOUI is with Mathematics and Computer Science Department, University of Tahar Moulay Saida, BP 138 ENNASR Saida 20000, Algeria, phone: 2130552967661; e-mail: b.sidaoui@gmail.com.
Kaddour SADOUNI is with Computer Science Department University of Sciences and Technology USTO-MB, BP 1505 Elmanouar Oran 31000, Algeria (kaddour_sadouni@hotmail.com).

introduce our efficient framework for multiclass SVM and we discuss the implementation of our architecture OVA Tree Multiclass. In section 5, we give results of experiments on the vowels sets of TIMIT data base and MNIST corpus. Finally, the last section is devoted to conclusions and some remarks pertaining to future work.

II. Support Vector Machines

The main idea of binary SVMs is to implicitly map data to a higher dimensional space via a kernel function and then solve an optimization problem to identify the maximum-margin hyper-plane that separates training instances [2]. The separator is based on a set of boundary training instances (training examples). Kernels can be interpreted as dissimilarity measures of pairs of objects in the training set X. In standard SVM formulations, the optimal hypothesis sought is of the form (1).

$$\Phi(\xi) = \sum \alpha_i k(x, x_i) \tag{1}$$

Where α_i are the components of the unique solution of a linearly constrained quadratic programming problem, whose size is equal to the number of training patterns. The solution vector obtained is generally sparse and the non zero α_i 's are called support vectors (SV's). Clearly, the number of SV's determines the query time which is the time it takes to predict novel observations and subsequently, is critical for some real time applications such as speech recognition.

The training process is implicitly performed in a Reproducing Kernel Hilbert Space in which k(x;y) is the inner product of the images of two example x, y. Moreover, the optimal hypothesis can be expressed in terms of the kernel that can be defined for non Euclidean data such biological sequences, speech utterances etc. Popular positive kernels include the Linear (2), Polynomial (3), and Gaussian (4), kernels:

$$k(x_i, x_j) = x_i^T x_j \tag{2}$$

$$k(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0 \tag{3}$$

$$k(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2), \gamma > 0 \tag{4}$$

A. SVM Formulation

Given training vectors $x_i \in \Re^n, i = 1, \ldots, m$, in two classes, and a vector $y \in \Re^m$ such that $y_i \in \{1, -1\}$, Support Vector Classifiers [2], [3], [4], [5] solve the following linearly constrained convex quadratic programming problem:

$$\begin{aligned} &\text{maximize } W(\alpha) = \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) - \sum_{i=1}^m \alpha_i \\ &\text{under the constraints:} \quad \forall i,\ 0 \le \alpha_i \le C \\ &\qquad \sum_{i=1}^m \alpha_i y_i = 0 \end{aligned} \tag{5}$$

The optimal hypothesis is:

$$f(x) = \sum_{i=1}^m \alpha_i y_i k(x, x_i) + b \tag{6}$$

Where the bias term b can be computed separately [6]. Clearly, the hypothesis f depends only on the non null coefficients α_i whose corresponding patterns are called Support vectors (SV). The quadratic programming (QP) objective function involves the problem Gram matrix K whose entries are the similarities $k(x_i, x_j)$ between the patterns x_i and x_j. It is important to note, on one hand, that the pattern input dimension d, in the above formulation, is implicit and does not affect to some extent the complexity of training, provided that the Gram matrix K can be efficiently computed for the learning task at hand. On the other hand, the patterns representation is not needed and only pair wise similarities between objects must be specified.

This feature makes SVM very attractive for high input dimensional recognition problems and for the ones where patterns can't be represented as fixed dimensional real vectors such as text, strings, DNA etc. For large scale corpora however, the quadratic programming problem becomes quickly computationally expensive, in terms of storage and CPU time. It is well known that general-purpose QP solvers scale with the cube of the problem dimension which is, in our case, the number of training patterns m. Specialized algorithms, typically based on gradient descent methods, achieve impressive gains in efficiency, but still become impractically slow for problems whose size exceeds 100,000 examples. Several attempts have been made to overcome this shortcoming by using heuristically based decomposition techniques such as Sequential Minimal Optimization [6] implemented in LibSVM package [22].

B. Extension Multiclass and Related Works

Many real–world applications consist of multiclass classification problems. Unfortunately, the Support Vector Machine (SVM), which is now considered one of the state–of–the–art algorithms in machine learning research, is intrinsically bi-class and its efficient extension to multiclass problems is still an ongoing research issue [23], [24], [25]. Several frameworks have been introduced to extend SVM to multiclass contexts and a detailed account of the literature is out of the scope of this paper.

The most common way to build a multiclass SVM is to combine several subproblems that involve only binary classification. This idea is used by various approaches as One-Versus -All (OVA), One-Versus-One (OVO) [2], [23] and Directed Acyclic Graph (DAGSVM) [24]: in the first case, K binary classifiers are constructed, where K is the number of classes. The k^{th} classifier is trained by labeling all the examples in the k^{th} class as positive and the remainder as negative. The final hypothesis is given by the formula:

$$f_{ova}(x) = \arg\max_{i=1,\dots,k}\left(f_i(x)\right) \qquad (7)$$

In the second case, OVO proceeds by training k(k-1)/2 binary classifiers corresponding to all the pairs of classes. The hypothesis consists of choosing either the class with most votes (voting). In the third case, each node of graph represents a binary classifier (DAGSVM) [24] [38].

There was debate on the efficiency of multiclass methods from statistical point of view clearly, voting and DAGSVM are cheaper to train in terms of memory and computer speed than OVA. [25] Investigated the performance of several SVM multi-class paradigms and found that the one-against-one achieved slightly better results on some small to medium size benchmark data sets. Furthermore, other interesting implementations of SVM multi-classes have been developed in recent years. Such as, two architectures proposed by [27] and [28], in the first Cha and Tappert built the decision tree by genetic algorithms where they are considered each tree as chromosome. While in the second architecture Madzarov and All propose a clustering method to construct the binary tree. Another approach in this direction has been presented by Lei and Venu in [29], they use a recursive method in to build the binary tree.

In term of complexity, OVA approach needs to create k binary classifiers; the training time is estimated empirically by a power law [30] stating that $T \approx \alpha M^2$ where M is the number of training samples and α is proportionality constant. According to this law, the estimated training time for OVA is:

$$T\_Time_{OVA} \approx k\alpha M^2 \qquad (8)$$

Without loss of generality, let's suppose that each of the k classes has the same number of training samples. Therefore, each binary SVM-(OVO) approach requires $2M/k$ samples. Hence, the training time for OVO is:

$$T\_Time_{OVO} \approx \alpha \frac{k(k-1)}{2}\left(\frac{2M}{k}\right)^2 \approx 2\alpha M^2 \qquad (9)$$

The training time for DAG is same as OVO. In the training phase, our framework One-Versus-All Tree multiclass has (k-1) binary classifiers (k is the number of classes). The random structure of the optimal tree complicates the calculation of the training time. However, an approximation is defined as: Let's assume that each of the k classes has the same number of training samples. The training time is summed over all k-1 nodes in the different $\lceil \log_2(k) \rceil$ levels of tree. In the i^{th} level, there are at the most 2^{i-1} nodes and each node uses $M/2^{i-1}$ training samples. Hence, the total training time:

$$T\_Time_{SVMAG} \approx \sum_{i=1}^{\lceil \log_2(k) \rceil} \alpha 2^{i-1}\left(\frac{M}{2^{i-1}}\right)^2 \approx \alpha M^2 \qquad (10)$$

It must be noted that the training time of our approach does not include the time to build the hierarchy structure (binary tree) of the k classes. In the testing phase, OVA require k binary SVM evaluations and OVO necessitate $\frac{k(k-1)}{2}$ binary SVM evaluations, while DAGSVM performs faster than OVO and OVA, since it requires only k-1 binary SVM evaluations. The two architectures proposed by [27], [35] and [28] and One-Versus-All Tree multiclass proposed in this paper are even fasters than DAGSVM because the depth of the binary tree is $\lceil \log_2(k) \rceil$. In addition, the total number of supports (SVs) vectors in all models will be smaller than the total number of SVs in the others (OVA, OVO and DAGSVM). Therefore it allows converge rapidly in the test phase.

The advantage of the approaches presented in [27], [28], [29] and the approach shown in this paper lie mainly in the test phase, because it uses only the models necessary for recognition. Which make the testing phase faster. However, in [27], [28] and [29] a problem of local minima is clearly. To avoid this problem, the work presented in [35] and this approach proposed in this work is to find the binary tree, using the similarity idea to find the right partitioning into two disjoint groups (one against rest), the partial optimization avoids falling into a local optimum, the details of the algorithm is discussed in the next section.

III. Binary Tree Multiclass SVM

This approach uses multiple SVMs set in a binary tree structure [21]. In each node of the tree, a binary SVM is trained using two classes. All samples in the node are assigned to the two subnodes derived from the current node. This step repeats at every node until each node contains only samples from one class. That said, until the leaves of the tree. The main problem that should be considered seriously here is how to construct the optimal tree? With the aim of partitioning correctly the training samples in two groups, in each node of the tree. In this sense, many works have been proposed in Literatures as [27, 28, 29 and 35]. In this paper, we introduce a new multiclass method, which we call One-versus-All Tree (OVA Tree Multiclass) for solving multiclass problems with a Support Vector Machine.

A. One- Versus –All Binary Tree

In this paper, we exploit the main idea of the OVA approach and criteria's of similarity. Binary tree constructing is based on the research of the best hyperplan separating a class against the rest by using two criteria's of similarity. The aim is to construct a binary tree where each node represents a partition of two classes (one class against the remains). While the leaves represent the class's labels.

But for a fixed number of classes, the optimal partition is not necessarily unique.

In this paper, we study several criteria specific to each family by systematically applying the same optimization strategy: a number of classes is fixed and, at each step, we construct a binary partition in each node of binary tree. At each level of the tree, we try to build a partition fixed number of classes (equal to 2) that optimizes a certain criterion. In this work, we begin with a partition has one group containing all classes. So we start with root (up) of tree, and we looks for the class of larger diameter (farthest compared to the other). We use the same procedure until no partitioning is possible. There are three families of natural classification criteria; it is separation, homogeneity and dispersion.

According to the first, a good score this well-separated classes, we seek to maximize the differences between classes, which are functions of distances between classes. According to the second, the classes are as concise as possible; it is desired to minimize the diameter, that is to say the maximum intra-distances classes. According to the third, we minimize a function of inertia, the sum of squared deviations in a center, whether real or virtual.

The OVA Tree Multiclass method that we propose is based on recursively partitioning the classes in two disjoint groups (one contains one class only) in every node of the binary tree, and training a SVM that will decide in which of the groups the incoming unknown sample should be assigned. In the general case, the number of partitions into two parts (groups) of a set of k elements is given by the following formula [26]:

$$N\_partions_{k,2} = \sum_{i=0}^{2} (-1)^i \frac{(2-i)^k}{i!(2-i)!} \qquad (11)$$

Corresponding construction of the binary tree, two cases can be expected: the number k is small in this case; we calculate all possible partitions and then deduce the optimal partition. Where the number k is greater than 6 (k> 6), we determine the optimal partition by Methods of optimization, because it is impossible to cover all possible partitions.

In this manuscript, we are interested at the first case. The OVA Tree two Algorithms are begun by a set X of training samples labelled by $y_i \in \{c_1, c_2, ..., c_k\}$, each OVA Tree Multiclass is summarized in two phases: Calculate k gravity centers and find the right partition of k gravity centers into two groups, by using separation, homogeneity criteria's. For that, a function of overall inertia is calculated for each binary tree. Indeed, this function of overall inertia is sum of inertia's of various partitions of tree. The figure (1) illustrates an example of binary tree for 7 classes.

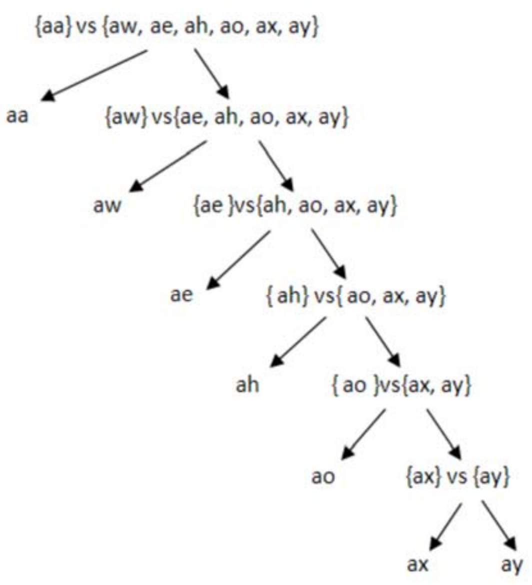

Fig. 1. Binary tree for 7 vowels

B. *Implementation of Tree SVM*

The two optimal trees are implemented to obtain a multiclass approaches. For each one, we take advantage of both the efficient computation of the tree architecture and the high classification accuracy of SVMs Utilizing this architecture, k-1 SVMs needed to be trained for k class problem. This binary tree is used to train a SVM classifier in the root node of the tree, using the samples of the first group as positive examples and the samples of the second group as negative examples. The classes from the first clustering group are being assigned to the left subtree, while the classes of the second clustering group are being assigned to the right subtree. The process continues recursively (dividing each of the groups into two subgroups applying the procedure explained above), until there is only one class per group which defines a leaf in the decision tree.

The recognition of each test sample starts at the root of the tree. At each node of the binary tree a decision is being made about the assignment of the input pattern into one of the two possible groups represented by transferring the pattern to the left or to the right sub-tree. Each of these groups may contain multiple classes. This is repeated recursively downward the tree until the sample reaches a leaf node that represents the class it has been assigned to.

IV. Experimental and Results

In this paper, we performed our experiments on two corpuses: TIMIT corpus [31] and MNIST corpus [39]. For TIMIT datasets, 20 vowels used in [32] are selected to evaluate our approach. The 20 vowels set are: {aa, aw, ae, ah, ao, ax, ay, axr, ax-h, uw, ux, uh, oy, ow, ix, ih, iy, eh, ey, er}. The 10 classes (handwritten digits) of MNIST corpus are: {0, 1, 2, 3, 4, 5, 6, 7, 8, and 9}.

In all the experiments reported below, we performed cross validation for tuning SVM hyper parameters. The GNU SVM light [22] implementation is used for our OVA Tree Multiclass Machine used in this paper, and LibSVM [33] for SVC [34] software's to compare our results. All the experiments were run on standard Intel (R) core™ 2 Duo

CPU 2.00 GHZ with 2.99 Go memory running the Windows XP operating system. The following tables, Table 1 and Table 2, summarize the preliminary results, for the classification supervised of 20 vowels, and Table 3 present results for 10 handwritten digits of MNIST listed above.

TABLE I: RESULTS OBTAINED BY SVM (OVO) FOR 20 VOWELS

	C	g	Test (%)	CPU time (s)
SVM (OVO)	2000	0.0005	59.64	510.89
	5000	0.0005	59.83	670.15
	10000	0.0005	60.14	504.00
	1000	0.005	58.11	945.65
	200	0.005	59.82	938.34

C is a parameter of SVM, g is Gaussian kernel parameter, the two parameters were calculated by cross validation.
Test is recognition rate and CPU time is time of test.

TABLE II: RESULTS OBTAINED BY OVA TREE MULTICLASS FOR 20 VOWELS OF TIMIT DATA SETS

	T	C	g	Test (%)	CPU Time (s)
Separation	0	0	-	40.80	120
	2	100	0.005	56.93	504
Homogeneity	2	0.0075	10	54.95	537
	2	0.0075	0	48.73	609
	2	0.005	10	53.62	572
	2	0.005	100	56.92	543

T is kernel type: Gaussian (2), linear (0).
C is a parameter of SVM, g is Gaussian kernel parameter, the two parameters were calculated by cross validation.
Test is recognition rate and CPU time is time of test.

TABLE III: RESULTS OBTAINED BY OVA TREE MULTICLASS FOR 10 DIGITS OF MNIST DATA SETS

	T	C	d	Test (%)	CPU Time (s)
Separation	0	0	-	91.69	300
	1	0	2	97.08	480
	1	10	2	97.73	420
	1	100	2	97.73	421
Homogeneity	0	0	-	91.75	264
	1	0	2	97.09	467
	1	100	2	97.72	418
	1	1000	2	97.72	420

T is kernel type: Polynomial (1), linear (0).
C is a parameter of SVM, d is Polynomial kernel parameter, the two parameters were calculated by cross validation.
Test is recognition rate and CPU time is time of test.

V. CONCLUSIONS

We introduced and implemented an efficient framework for SVM multiclass using separation and homogeneity criteria's. The Binary Tree of support vector machine (SVM) multiclass paradigm was shown through extensive experiments to achieve state of the art results in terms of accuracy. The preliminary experiments of our binary architecture named OVA Tree Multiclass indicate that the results are comparable to those of other methods, particularly [23] and [35] who used the same corpus. Nevertheless, parameters SVM optimization can improve the performance of our contribution.

However, we believe that in order to improve automatic pattern recognition technology, more research efforts must take advantage of the solid mathematical basis and the power of SVM binary, and should be invested in developing general purpose efficient machine learning paradigms capable of handling large scale multi-class problems.

REFERENCES

[1] Boser, B. Guyon, V Vapnik, "A training algorithm for optimal margin classifiers". *Fifth Annual Workshop on Computational Learning Theory*. ACM Press, Pittsburgh, 1992.
[2] V. Vapnik, "Statistical Learning Theory", Wiley, New York, 1998.
[3] Guyon, I. Boser and V. Vapnik, "Automatic Capacity Tuning of Very Large VC-Dimension Classifiers"; Advances in Neural Information *Processing Systems,* Vol.5 Morgan Kaufmann, San Mateo, CA, 1993.
[4] V. Vapnik and Chervonenkis, "Theory of Pattern Recognition [in Russian]". Nauka, Moscow, 1974. (German Translation: Wapnik, Tscherwonenkis, Theorie der Zeichenerkennung, Akademie-Verlag, Berlin), 1979.
[5] V. Vapnik, 1982. "Estimation of Dependences Based on Empirical Data [in Russian]. Nauka, Moscow, (English translation, Springer Verlag, New York), 1979.
[6] E. Osuna and all, "Training Support Vector Machines, an Application to Face Detection", *in Computer Vision and Pattern Recognition*, pp.130-136, 1997.
[7] T. Joachims, "Making large-scale support vector machine learning practical", *in Advances in Kernel Methods*, B. Schölkopf, C. Burges, A. Smola, (eds.), Cambridge, MIT Press, 1998.
[8] Daniel Hong, "Speech recognition technology: moving beyond customer service". *Computer Business Online*, March 1st, 2007.
[9] K-F. Lee and H-W, "Hon. Speaker-independent phone recognition using Hidden Markov Models"; I*EEE Trans, Acoust, Speech Signal Processing*, ASSP-37, N°11, 1989.
[10] Fei Sha and Lawrence K. Saul, "Large Margin Hidden Markov Models for Automatic speech recognition", *NIPS*, 2006.
[11] Fei Sha and all "Comparaison of Large Margin Training To Other Discriminative Methods for Phonetic Recognition by Hidden Markov Models", *ICASSP*, 2007.
[12] Hakan Erdogan, "Regularizing Linear Discriminant Analysis for Speech Recognition", *Interspeech'2005*, Lisbon, Portugal, September 4-8, 2005.
[13] F-Ghinwa Choueiter and R-James Glass, "A Wavelet and Filter Bank Framework for Phonetic Classification", *ICASSP*, 2005.
[14] M. Kamal and H-J. Mark, "Non-Linear Independent Component Analysis for Speech Recognition", *International Conference on Computer Communication and Control Technologies*, Orlando, 2003.
[15] H. Naomi, V. Saeed and MC. Paul, "A Novel Model for Phoneme Recognition Using Phonetically Derived features", *Proceeding EUSIPCO*, 1998.
[16] S. Furui. "Speaker-Independent Isolated Word Recognition Using Dynamic Features of Speech spectrum"; *IEEE Trans, Acoustic, Speech, and Signal Processing* 34, 52-59.
[17] P. Moreno, "On the use of Support Vector Machines for Phonetic Classification"; *proceedings of ICCASP*, 1999.
[18] Alex Grave and Jürgen Schmidhuber, "Framewise Phoneme Classification with Bidirectional LSTM Networks". *IJCNN*, 2005.
[19] J. Morris and E. Fosler-Lussier, "Discriminative Phonetic Recognition with Conditional Random Fields"; *HLTNAACL*, 2006.
[20] J. Salomon, k. Simon and Miles Osborne, "Framewise Phone classification Using Support Vector Machines", *ICSLP*, 2002.
[21] B. Fei, J. Liu. Binary Tree of SVM: "A New Fast Multiclass Training and Classification Algorithm", *IEEE Transaction on Neural networks*, Vol17, N°3, 2006.N13
[22] T. Joachims, "Making Large-Scale SVM Learning Practical", Software available at: *http://svmlight.joachims.org/*, 2001.
[23] Ryan Rifkin and Aldebaro Klautau, "In defense of one-vs-all classification", *Journal of Machine Learning Research* 5, 101-141, 2004.
[24] J.C. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin DAGs for multiclass classification"; *In Advances in Neural Information Processing Systems*, volume 12, pages 547-443. MIT Press, 2000.
[25] Chih-Wei Hsu and Chih-Jen Lin. "A Comparison of Methods for Multiclass Support Vector Machines", *IEEE Transactions on Neural Networks*, 2002.
[26] Jain, A. and Dubes, R, "Algorithms for Clustering Data". *Prentice Hall Advanced, Reference Series*, 1988.

[27] Sung-Hyuk Cha and Charles Tappert. "A Genetic Algorithm for Constructing Compact Binary Decision Trees". *Journal of Pattern Recognition Research* 1, 1-13, 2009.

[28] Gjorgji Madzarov, Dejan Gjorgjevikj and Ivan Chorbev. "A Multi-class SVM Classifier Utilizing Binary Decision Tree". Informatica 33, 233-241, 2009.

[29] Hansheng Lei and Venu Govindaraju. "Half-Against-Half Multi-class Support Vector Machines", Multiple Classifier Systems 2005: 156-164.

[30] J. Platt. Fast training of support vector machines using sequential minimal optimization. In Advances in Kernel Methods - Support Vector Learning.

[31] M. Slaney, Auditory Toolbox version 2. Tech. Report#010, Internal Research Corporation, 1998.

[32] R. Rifkin and a1, "Noise Robust Phonetic Classification with Linear Regularized Least Squares and Second Order Featues", ICASSP, 2007.

[33] C-C. Chang and C-J. Lin, "LIBSVM Toolkit: a library for support vector machines", Software available at: http://www.csie.ntu.edu.tw/cjlin/libsvm. 2001.

[34] LibCVM Toolkit of the improved Core Vector Machine (CVM), which are fast Support Vector Machine (SVM) training algorithms using core-set approximation on very large scale data sets available at: *http://c2inet.sce.ntu.edu.sg/ivor/cvm.html*.

[35] B. Sidaoui and K. Sadouni "Approach Multiclass SVM Utilizing Genetic Algorithms". ICDMA_10, 2013 *(submitted and accepted)*.

[36] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. *In Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, 1992.

[37] V. Vapnik. The Nature of Statistical Learning Theory. Springer, New York, 1995. B. Schölkopf, A. J. Smola, *Learning with Kernels*, MIT Press, 2002.

[38] J.C. Platt, N. Cristianini, J. Shawe–Taylor, "Large Margin DAGs for Multiclass Classification", *in Advances in Neural Information Processing Systems*, Vol. 12, pp. 547–553, The MIT Press, 2000.

[39] The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centred in a fixed-size image available at: *http://yann.lecun.com/exdb/mnist/*.