Approach Multiclass SVM Utilizing Genetic Algorithms

Boutkhil Sidaoui, Kaddour Sadouni

Abstract- **In this paper, we investigate and evaluate the performance of a simple approach mult[1]iclass for support vector machine (SVM) method. We present a new architecture, named GASVM, and based en genetic algorithms, in order to achieve high classification efficiency for multiclass problems. The proposed paradigm builds a binary tree, for solving multiclass problems, by genetic algorithms with the aim of obtaining a strategy multiclass SVM with a reasonable and practical complexity in the real problems. Our approach is more accurate in the creation of the tree. Further, in the test phase our contribution, due to its Log complexity, it is much faster than other methods in problems that have big class number. For the evaluation two corpuses are used; TIMIT corpus, where we achieved a recognition rate of 57.54% on the 20 vowels and MNIST datasets who's a recognition rate of 97.73% is achieved. These results are comparable with the state of the arts. 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 tasks.**

Keywords: **Machine Learning; SVM; Genetic algorithms.**

I. Introduction

KERNEL Methods and particularly Support Vector Machines (SVM) [14,15,16], introduced during the last decade in the context of statistical learning [15,17,18], have been successfully used for the solution of a large class of machine learning tasks [19,20] such as categorization, prediction, novelty detection, ranking and clustering. The 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. The popular 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 SVMs.

The paper is organized as follows. In section 2, support vector machine will be briefly discussed for problems classification. In the following section, we introduce our efficient framework for multiclass SVM using genetic algorithms. We discuss some related successful works in sections 4. In section 5, we give results of preliminary experiments on the vowels sets of TIMIT data base and digits sets of MNIST corpus. Finally, the last section is devoted to conclusions and some remarks pertaining to future work.

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).

II. Support Vector Machine

Binary SVM, in their general form, extend an optimal linear hypothesis, in terms of an upper bound on the expected risk that can be interpreted as the geometrical margin , to non linear ones by making use of kernels k(.,.). 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 tasks.

It is worth noting that in contrast to connectionist methods such as neural networks, the examples need not have a Euclidean or fixed-length representation when used in kernel methods. The training process is implicitly performed in a Reproducing Kernel Hilbert Space (RKHS) 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, Polynomial and Gaussian kernels:

A. SVM Formulation

Given training vectors $x_i \in \Re^n, i = 1,...,m$, in two classes, and a vector $y \in \Re^m$ such that $y_i \in \{1,-1\}$, Support Vector Classifiers [15,16,17,18] solve the following linearly constrained convex quadratic programming problem:

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

The optimal hypothesis is:

$$f(x)=\sum_{i=1}^{m}\alpha_i y_i k(x,x_i)+b \qquad (3)$$

Where the bias term b can be computed separately [19]. Clearly, the hypothesis f depends only on the non null coefficients α_i whose corresponding patterns are called Support vectors (SV).

The 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 SMO [19] implemented in LibSVM package [26].

B. Multiclass Extensions

Support Vector Machines are inherently binary classifiers and its efficient extension to multiclass problems is still an ongoing research issue [22, 23, 24]. 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. Typically multiclass classifiers are built by combining several binary classifiers. The earliest such method is the One-Against-All (OVA) [15, 22] which constructs K classifiers, 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:

$$f_{ova}(x)=\arg\max_{i=1,\ldots\ldots,k}\left(f_i(x)\right) \qquad (4)$$

Another popular paradigm, called One-Against-One (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) or traversing a directed acyclic graph where each node represents a binary classifier (DAGSVM) [23]. 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 OVASVM .[24] 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. Other interesting works will be discussed in section: related work and discussion.

III. Binary Tree for Multiclass SVM

This approach uses multiple SVMs set in a binary tree structure [40]. 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 paper we propose a genetic algorithm for constructing a binary tree structure for multiclass SVM.

A. Binary Tree Construction

Genetic algorithms (GA) can provide good solutions to many optimization problems. They are based on natural processes of evolution. The process of genetic algorithm is defined as follows: coding, selection, genetic operators such as mutation and crossover. The GASVM method that we propose is based on recursively partitioning the classes in two disjoint groups 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. The groups are determined by genetic algorithm.

In the general case, the number of partitions into two parts (groups) of a set of k elements is given by the following formula [46]:

$$N\_partions_{k,2}=\sum_{i=0}^{2}(-1)^i\frac{(2-i)^k}{i!(2-i)!} \qquad (5)$$

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 genetic algorithms, because it is impossible to cover all possible partitions.

B. Preliminary

Let's take a set of training samples labeled by $y_i\in\{c_1,c_2,\ldots,c_k\}$, GASVM method starts, in the first step, with calculating k gravity centers for the k different classes. In the second step, the goal is to find the right partitioning of k gravity centers into two disjoint groups. For that, a process of genetic algorithm is applied for each node of the binary tree as follow: we start from the root of the binary tree, we partition the k gravity centers (k classes) into two groups is the most disjointed, says the chromosome having the maximum fitness function with genetic algorithms. Then, we continue this operation for nodes derived until no partitioning is possible, that is to say nodes are the leaves of the tree. After that, a function of overall inertia is calculated for each binary tree (individual). Indeed, this function of overall inertia is given by the following formula:

$$f_{overallinertia} = \sum_{i=1}^{nodes} fun\_fiteness(i) \qquad (6)$$

Where, $fun\_fiteness(i)$ is the fitness function of the node i.

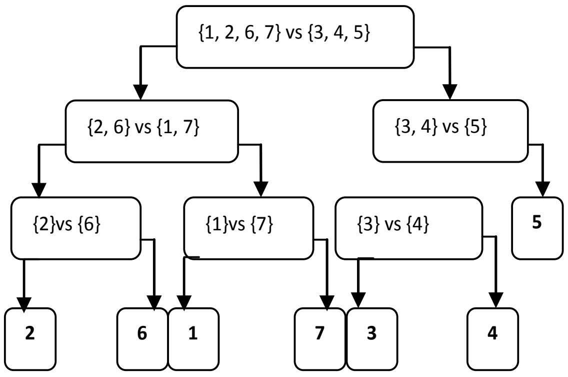

Fig. 1. shows an example binary tree for seven classes

C. Genetic algorithms for constructing optimal partition

Encoding: For genetic algorithms to build optimal partition, partition must be encoded so that genetic operators such as mutation and crossover can be applied. Let $y_i \in \{c_1, c_2, ..., c_k\}$ be the label list of k gravity centers for the k different classes. For encoding each partition, permutation encoding is used. In permutation encoding, every chromosome is a string of numbers. In this paper, each number represents a vowel; therefore partition is encoded by a string of numbers. In this paper we used the set: {aa, ae, ah, ao, aw, ax-h, ax, axr, ay, eh, er, ey, ih, ix, iy, ow, oy, uh, uw, ux} for datasets TIMIT vowels. The vowels are represented by numbers 1 to 20 respectively and for MNIST datasets each digit is represented by number.

TABLE I: EXAMPLE OF CHROMOSOMES (PARTITIONS)

Chromosome 1	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Chromosome 2	13 12 3 4 5 6 7 8 9 10 11 2 1 14 15

Chromosome 1, Chromosome 2: two examples of chromosome representation (digit sequence).

Fitness Function: fitness function of each partition is the Euclidean distance between the two groups that partition.

Genetic Operators: Crossover and mutation are two basic operators of GA. Performance of GA very depend on them. Type and implementation of operators depends on encoding and also on a problem. The Crossover selects genes from parent chromosomes and creates a new offspring. Whereas, mutation operator is defined as changing the value of a certain position in a string to one of the possible values in the range. There are many ways how to do crossover and mutation. In this paper, we are only interested by crossover to a single point, Single point crossover is selected, till this point the first part is copied from the first parent, and then the rest is copied from the second parent. For mutation, change in rank; two numbers are selected and exchanged.

TABLE II: EXAMPLE OF CROSSOVER WITH A SINGLE POINT

Partition 1	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Partition 2	13 12 3 4 5 6 7 8 9 10 11 2 1 14 15
Offspring 1	1 2 3 4 5 6 7 8 9 **10 11 2 1 14 15**
Offspring 2	**13 12 3 4 5 6 7 8 9** 10 11 12 13 14 15

Partition 1, Partition 2: two fathers, after crossover with a single point (at 9 point), we obtain the two representations: Offspring 1 and 2.

But the problem is that some numbers are repeated and others are missing, while all numbers are included in each chromosome. For this, correction is applied to the offspring to add the missing number and remove duplication.

TABLE III: EXAMPLE OF MUTATION OPERATOR

Before mutation	1 2 3 4 **5** 6 7 8 9 10 11 12 13 14 **15**
After mutation	1 2 3 4 **15** 6 7 8 9 10 11 12 13 14 **5**

Before mutation: example of chromosome representation.
After mutation: chromosome after mutation operator of a single digit (5 by 15).

Decoding: Decoding is the reverse operator of the encoding process. In this paper the best individual is the tree with the maximum overall inertia function.

D. Implementation of Tree SVM

The optimal tree (best individual) created by genetic algorithms is implemented to obtain a multiclass approach. Takes advantage of both the efficient computation of the tree architecture and the high classification accuracy of SVMs Utilizing this architecture, N-1 SVMs needed to be trained for an N 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 first (left) subtree, while the classes of the second clustering group are being assigned to the (right) second 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. Related Work and Discussion

There are different approaches for solving multi-class problems which are not based on SVM. However, the experimental results clearly show that their classification accuracy is significantly smaller than the SVM based methods.

Several multi-class approaches have been proposed to solve the multi-class problem for SVM. SVM using the standard formulation cannot deal directly multi-class problems. Early implementations attempted to treat the multiclass problem are the OVA and the OVO approaches already discussed in Section 2. Most of the recent works are based on the combination of classifiers binaries.

Interesting implementations of SVM multi-classes have been developed in recent years. Such as, two architectures proposed by [42] and [41], 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 proposed a clustering method to construct the binary tree. Another approach in this direction has been presented by Lei and Venu in [43], 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 [44] 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:

$$Training\_Time_{OVA} \approx k\alpha M^2 \tag{7}$$

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

$$Training\_Time_{OVO} \approx \alpha \frac{k(k-1)}{2}\left(\frac{2M}{k}\right)^2 \approx 2\alpha \frac{(k-1)}{k} M^2 \approx 2\alpha M^2 \tag{8}$$

The training time for DAG is same as OVO. In the training phase, our approach GASVM 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:

$$Training\_Time_{SVMAG} \approx \sum_{i=1}^{\lceil \log_2(k) \rceil} \alpha 2^{i-1}\left(\frac{M}{2^{i-1}}\right)^2 \approx \sum_{i=1}^{\lceil \log_2(k) \rceil} \alpha \left(\frac{M^2}{2^{i-1}}\right) \approx \alpha M^2 \tag{9}$$

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 [42] and [41] and GASVM 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 [42], [41], [43] 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 [42], [41] and [43] a problem of local minima is clearly present. To avoid this problem, an approach proposed in this works to find the binary tree, using the process of genetic algorithm to each node of the tree to find the right partitioning into two disjoint groups, the partial optimization avoids falling into a local optimum, the details of the algorithm is discussed in section 5 above.

V. Implementation and Results

In this paper, we performed our experiments on two corpuses: TIMIT corpus [30] and MNIST corpus [47]. For TIMIT datasets, 20 vowels used in [36] 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 5-fold cross validation for tuning SVM hyper parameters g and C. The GNU SVM light [26] implementation is used for our GASVM Machine used in this paper, and LibSVM [45] for SVC [39] 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 summarize our preliminary results, for the classification supervised of 20 vowels listed above.

TABLE IV: RESULTS OF SVM (OVO) FOR 20 VOWELS

	T	C	g	Test (%)	CPU time (s)
SVM (OVO)	2	2000	0.0005	59.64	510.89
	2	5000	0.0005	59.83	670.15
	2	10000	0.0005	60.14	504.00
	2	1000	0.005	58.11	945.65
	2	200	0.005	59.82	938.34

T is kernel type: Gaussian (2).
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 V: RESULTS OF GASVM FOR 20 VOWELS

	T	C	g	Test (%)	CPU time (s)
GASVM	2	4	0.03	57.16	534
	2	10	0.03	56.23	584
	2	5	0.03	57.29	537
	2	100	0.007	57.32	482
	2	100	0.01	56.83	520
	2	100	0.0055	57.54	424

T is kernel type: Gaussian (2).
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 VI: RESULTS OF GASVM FOR 10 DIGITS

	T	C	d	Test (%)	CPU time (s)
GASVM	1	0	2	96.83	536
	0	0	-	90.86	250
	1	10	2	97.38	386
	1	100	2	97.38	341
	1	1000	2	97.73	406

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.

VI. CONCLUSION

We introduced and implemented a novel efficient approach for SVM multiclass. 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 indicate that the results are comparable to those of other methods. Although the optimization of learning parameters of SVM can improve the results.

However, as can be seen from the results, training times and especially testing times are still excessive preventing SVM from being used in speech recognition at least for the time being. We believe that in order to improve automatic speech 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] K-F. Lee and H-W, "Hon. Speaker-independent phone recognition using Hidden Markov Models"; *IEEE Trans, Acoust, Speech Signal Processing*, vol ASSP-37, N° 11, 1989.
[2] Alex Grave and Jürgen Schmidhuber, "Framewise Phoneme Classification with Bidirectional LSTM Networks". *IJCNN*, 2005.
[3] Hakan Erdogan, "Regularizing Linear Discriminant Analysis for Speech Recognition", 2005.
[4] J. Morris and E. Fosler-Lussier, "Discriminative Phonetic Recognition with Conditional Random Fields"; *HLTNAACL*, 2006.
[5] DongSuk. Yuk and James Flanagan, "Telephone Speech Recognition Using Neural Networks and Hidden Markov Models", *ICASSP*, 1999.
[6] F-Ghinwa Choueiter and R-James Glass, "A Wavelet and Filter Bank Framework for Phonetic Classification", *ICASSP*, 2005.
[7] K. Joseph, S. Shalev-Shwartz, S. Bengio, Y.Singer and D. Chazan, "Discriminative Kernel-Based Phoneme Sequence Recognition", *ICSLP*, 2006.
[8] Simon King, Todd Stephenson, Stephan Isard, Paul Taylor and Alex Strahan, "Speech Recognition via Phonetically Featured Syllables", ICSLP, 1998.
[9] Fei Sha and Lawrence K. Saul, "Large Margin Hidden Markov Models For Automatic speech recognition", *NIPS*, 2006.
[10] Fei Sha and Lawrence K. Saul, "Comparaison of Large Margin Training To Other Discriminative Methods for Phonetic Recognition by Hidden Markov Models", *ICASSP*, 2007.
[11] M. Kamal Omar and H-J. Mark, "Non-Linear Independent Component Analysis for Speech Recognition", *International Conference on Computer Communication and Control Technologies*, Orlando, 2003.
[12] H. Naomi, V. Saeed and MC. Paul, "A Novel Model for Phoneme Recognition Using Phonetically Derived features", Proceeding *EUSIPCO*, 1998.
[13] J. Salomon, k. Simon and Miles Osborne, "Framewise Phone classification Using Support Vector Machines", *ICSLP*, 2002.
[14] Boser, B. Guyon, V Vapnik, "A training algorithm for optimal margin classifiers". *Fifth Annual Workshop on Computational Learning* Theory. ACM Press, Pittsburgh, 1992.
[15] V. Vapnik, "Statistical Learning Theory", Wiley, New York, 1998.
[16] 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.
[17] V. Vapnik and Chervonenkis, "Theory of Pattern Recognition [in Russian]". Nauka, Moscow, 1974. (German Translation: Wapnik, Tscherwonenkis, Theorie der Zeichenerkennung, Akademie-Verlag, Berlin), 1979.
[18] V. Vapnik, 1982. "Estimation of Dependences Based on Empirical Data [in Russian]. Nauka, Moscow, (English translation, Springer Verlag, New York), 1979.
[19] E. Osuna, R. Freund, F. Girosi, "Training Support Vector Machines, an Application to Face Detection", *in Computer vision and Pattern Recognition*, pp.130-136, 1997.
[20] 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.
[21] Daniel Hong, "Speech recognition technology: moving beyond customer service". *Computer Business Online*, March 1st, 2007.
[22] Ryan Rifkin and Aldebaro Klautau, "In defense of one-vs-all classification", *Journal of Machine Learning Research* 5, 101-141, 2004.
[23] 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.
[24] Chih-Wei Hsu and Chih-Jen Lin. "A Comparison of Methods for Multiclass Support Vector Machines".
[25] S. Furui. "Speaker-Independent Isolated Word Recognition Using Dynamic Features of Speech spectrum"; *IEEE Trans, Acoustic, Speech, and Signal Processing* 34, 52-59.
[26] T. Joachims, "Making Large-Scale SVM Learning Practical", Software available at: *http://svmlight.joachims.org/, 2001*.
[27] A. Ganapathiraju, "Support vector machines for speech recognition". *PhD Thesis, Missisipi State University*, USA, 1998.
[28] N. Smith and M. Gales, "Speech recognition using SVM. Advances in Neural Information Processing Systems", 14, MIT Press, 2002.
[29] J.S. Garofolo, L.F. Lamel and al, "TIMIT Acoustic-Phonitic Continuous Speech Corpus. Philadelphia, Linguistic Data Consortium", 1993.
[30] M. Slaney, Auditory Toolbox version 2. Tech. Report#010, *Internal Research Corporation*, 1998.
[31] Ryan Rifkin and al, "Phonetic Classification Using Hierarchical, Feed-forward, Spectro-temporal Patch-based Architectures". *Technical Report, MIT-CSAIL-TR*, 2007.
[32] R. P. Lippmann, "Speech recognition by machines and humans", *Speech Communication* 22 1-15, *Elsevier*, 1997.
[33] Martens J. P. Depuydt, "Broad phonetic classification and segmentation of continuous speech by means of neural networks and

dynamic programming", *Speech communication*, vol. 10, no1, pp. 81-90, 1991.

[34] L. R. Rabiner and B-H Juang, "Fundamentals of speech recognition", *Prentice Hall*, 1993.

[35] P. Moreno, "On the use of Support Vector Machines for Phonetic Classification"; *In the proceedings of ICCASP*, 1999.

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

[37] J.C. Platt, "Fast training of support vector machines using sequential minimal optimization". *In Adv. in Kernel Methods*, Schölkopf, C. Burges, A. Smola eds, 1998.

[38] H. Peter Graf, E. Cosatto, B. Léon, I. Dourdanovic, and V. Vapnik, "Parallel Support Vector Machines: The Cascade SVM" *in Advances in Neural Information Processing Systems*, 2005.

[39] 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*.

[40] B. Fei, J. Liu. "Binary Tree of SVM: A New Fast Multiclass Training and Classification Algorithm" *IEEE Transaction on neural networks*, Vol. 17, No. 3, May 2006

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

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

[43] Hansheng Lei and Venu Govindaraju. "Half-Against-Half Multi-class Support Vector Machines".

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

[45] 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*.

[46] Jain, A. and Dubes, R, "Algorithms for Clustering Data". *Prentice Hall Advanced, Reference* Series, 1988.

[47] 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/*.