MULTI-CLASS CLASSIFICATION USING SUPPORT VECTOR MACHINES IN DECISION TREE ARCHITECTURE

Gjorgji Madzarov, Dejan Gjorgjevikj, *member , IEEE*

Abstract: **A novel architecture of Support Vector Machine classifiers utilizing binary decision tree (SVM-DTA) for solving multiclass problems is proposed in this paper. A clustering algorithm was used to determine the hierarchy of binary decision subtasks performed by the SVM binary classifiers. The applied clustering model utilizes Mahalanobis distance measures at the kernel space for better consistency with the used SVM kernel. The proposed SVM based Binary Decision Tree architecture takes advantage of both the efficient computation of the decision tree architecture and the high classification accuracy of SVMs. The performance of the proposed SVM-DTA was estimated on a problem of recognition of handwritten digits and letters. The experiments were conducted with samples from Pendigit and Statlog databases of segmented digits and letters. The results of the experiments indicate that the proposed method is faster to be trained than the other methods. Also, due to its Log complexity, the proposed SVM-DTA is much faster than the widely used multi-class SVM methods like "one-against-one" and "one-against-all", maintaining comparable accuracy. The experiments also showed that this method becomes more favorable as the number of classes in the recognition problem increases.**

Index Terms**: Support Vector Machine, multi-class classification, clustering, binary decision tree architecture.**

I. INTRODUCTION

The Recent results in pattern recognition have shown that support vector machine (SVM) classifiers often have superior recognition rates in comparison to other classification methods. However, the SVM was originally developed for binary decision problems, and its extension to multi-class problems is not straight-forward. How to effectively extend it to solve multi-class classification is still an on-going research issue. The popular methods for applying SVMs to multi-class classification problems usually decompose the multi-class problems into several two-class problems that can be addressed directly using several SVMs.

For the readers' convenience, we will introduce the SVM briefly in Section 2. A brief introduction to several widely used multi-class classification methods that utilize binary SVMs will be given in Section 3. The Kernel-based clustering introduced to convert the multi-class problem into SVM-based binary decision-tree architecture is explained in Section 4. In section 5, we discuss related works and compare SVM-DTA with other multi-class SVM methods via theoretical analysis and empirical estimation. The experimental results are presented to compare the performance of the proposed SVM-DTA with traditional multi-class approaches in Section 6. Section 7 gives a conclusion of the paper.

II. SUPPORT VECTOR MACHINES FOR PATTERN RECOGNITION

The support vector machine is originally a binary classification method developed by Vapnik and colleagues at Bell laboratories [1][2], with algorithm improvements by others [3]. For binary problem, we have training data points: $\{x_i, y_i\}, i = 1,...,l,$. $y_i \in \{-1,1\}, x_i \in R^d$. Suppose we have some hyperplane which separates the positive from the negative examples (a "separating hyperplane"). The points x which lie on the hyperplane satisfy $w \cdot x + b = 0$, where w is normal to the hyperplane, $|b|/\|w\|$ is the perpendicular distance from the hyperplane to the origin, and $\|w\| \leftarrow$ is the Euclidean norm of w . Let $d_+(d_-)$ be the shortest distance from the separating hyperplane to the closest positive (negative) example. Define the "margin" of a separating hyperplane to be $d_+ + d_-$. For the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin. This can be formulated as follows: suppose that all the training data satisfy the following constraints:

$$x_i \cdot w + b \geq +1 \text{ for } y_i = +1, \quad (1)$$

$$x_i \cdot w + b \leq -1 \text{ for } y_i = -1, \quad (2)$$

These can be combined into one set of inequalities:

$$y_i(x_i \cdot w + b) - 1 \geq 0 \quad \forall i, \quad (3)$$

Now consider the points for which the equality in Eq. (1) holds (requiring that there exists such a point is equivalent to choosing a scale for w and b). These points lie on the hyperplane $H_1 : x_i \cdot w + b = 1$ with normal w and perpendicular distance from the origin $|1-b|/\|w\|$. Similarly, the points for which the equality in Eq. (2) holds lie on the hyperplane $H_2 : x_i \cdot w + b = -1$, with normal again w , and perpendicular distance from the origin $|-1-b|/\|w\|$. Hence $d_+ = d_- = 1/\|w\| \leftarrow$ and the margin is simply $2/\|w\|$.

978-1-4244-3861-7/09/$25.00 ©2009 IEEE

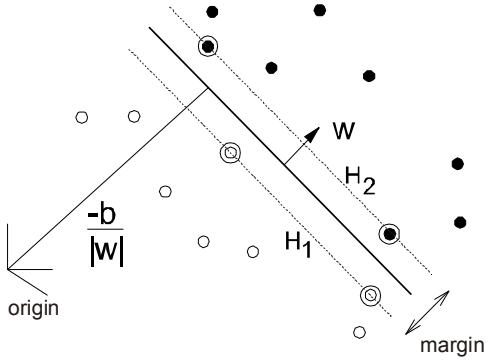

Fig. 1 – Linear separating hyperplanes for the separable case. The support vectors are circled.

Note that H_1 and H_2 are parallel (they have the same normal) and that no training points fall between them. Thus we can find the pair of hyperplanes which gives the maximum margin by minimizing $\|w\|^2$, subject to constraints (3).

Thus we expect the solution for a typical two dimensional case to have the form shown on Fig. 1. We introduce positive Lagrange multipliers $\alpha_i, i = 1,...,l,$, one for each of the inequality constraints (3). Recall that the rule is that for constraints of the form $c_i \geq 0$, the constraint equations are multiplied by *positive* Lagrange multipliers and subtracted from the objective function, to form the Lagrangian. For equality constraints, the Lagrange multipliers are unconstrained. This gives Lagrangian:

$$L_p \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{l} \alpha_i \ , \qquad (4)$$

We must now minimize L_p with respect to w, b, and simultaneously require that the derivatives of L_p with respect to all the α_i vanish, all subject to the constraints $\alpha_i \geq 0$ (let's call this particular set of constraints C_1). Now this is a convex quadratic programming problem, since the objective function is itself convex, and those points which satisfy the constraints also form a convex set (any linear constraint defines a convex set, and a set of N simultaneous linear constraints defines the intersection of N convex sets, which is also a convex set). This means that we can equivalently solve the following "dual" problem: *maximize* L_p, subject to the constraints that the gradient of L_p with respect to w and b vanish, and subject also to the constraints that the $\alpha_i \geq 0$ (let's call that particular set of constraints C_2). This particular dual formulation of the problem is called the Wolfe dual [4]. It has the property that the maximum of L_p, subject to constraints C_2, occurs at the same values of the w, b and α, as the minimum of L_p, subject to constraints C_1.

Requiring that the gradient of L_p with respect to w and b vanish give the conditions:

$$w = \sum_i \alpha_i y_i x_i \ , \qquad (5)$$

$$\sum_i \alpha_i y_i = 0 \ . \qquad (6)$$

Since these are equality constraints in the dual formulation, we can substitute them into Eq. (4) to give

$$L_D \equiv \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \ , \qquad (7)$$

Note that we have now given the Lagrangian different labels (P for primal, D for dual) to emphasize that the two formulations are different: L_p and L_D arise from the same objective function but with different constraints; and the solution is found by minimizing L_p or by maximizing L_D. Note also that if we formulate the problem with $b = 0$, which amounts to requiring that all hyperplanes contain the origin, the constraint (6) does not appear. This is a mild restriction for high dimensional spaces, since it amounts to reducing the number of degrees of freedom by one.

Support vector training (for the separable, linear case) therefore amounts to maximizing L_D with respect to the α_i, subject to constraints (6) and positivity of the α_i, with solution given by (5). Notice that there is a Lagrange multiplier α_i for every training point. In the solution, those points for which $\alpha_i > 0$ are called "support vectors", and lie on one of the hyperplanes H_1, H_2. All other training points have $\alpha_i = 0$ and lie either on H_1 or H_2 (such that the equality in Eq. (3) holds), or on that side of H_1 or H_2 such that the strict inequality in Eq. (3) holds. For these machines, the support vectors are the critical elements of the training set. They lie closest to the decision boundary; if all other training points were removed (or moved around, but so as not to cross H_1 or H_2), and training was repeated, the same separating hyperplane would be found.

For the nonlinear case, we may solve the problem in a certain high dimension space by a kernel map $\Phi(\cdot)$.

$$L_D \equiv \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j \Phi(x_i)\cdot\Phi(x_j), \qquad (8)$$

where $\Phi(x_i)\cdot\Phi(x_j) = k(x_i, x_j)$. That is, the dot product in that high dimension space is equivalent to a kernel function of the input space.

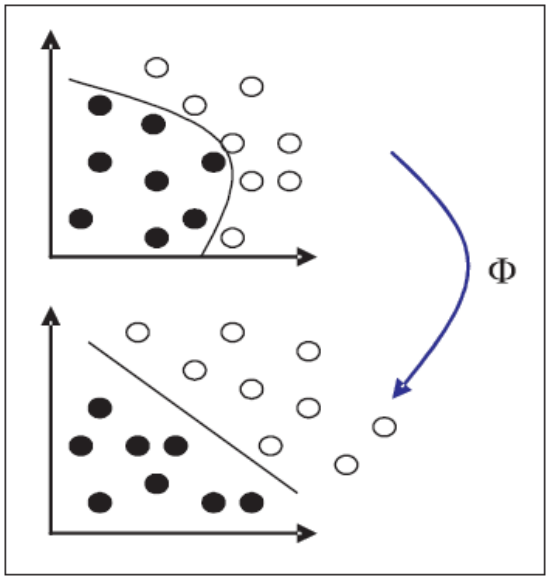

Fig. 2 – General principle of SVM: projection of data in an optimal dimensional space.

The Gaussian kernel is used in our experiments

$$K(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right), \qquad (9)$$

Gaussian is a reasonable first choice. The Gaussian kernel nonlinearly maps samples into a higher dimensional space, so unlike the linear kernel, it can handle the case when the relation between class labels and attributes is nonlinear.

For imperfect separation, the penalty C is used to penalize the data points that cross the boundaries. The only difference from the perfectly separating case is that α_i is bounded by C

$$\begin{cases} 0 \le \alpha_i \le C \\ \sum_i \alpha_i y_i = 0 \end{cases}. \qquad (10)$$

III. OVERVIEW OF WIDELY USED MULTI-CLASS CLASSIFICATION METHODS

Although SVMs were originally designed as binary classifiers, approaches that address a multi-class problem as a single "all-together" optimization problem exist [5], but are computationally much more expensive than solving several binary problems.

A variety of techniques for decomposition of the multi-class problem into several binary problems using Support Vector Machines as binary classifiers have been proposed, and several widely used are:

A. *One-against-all(OvA)*

For the N-class problems $(N > 2)$, N 2-class SVM classifiers are constructed [6]. The i^{th} SVM is trained while labeling the samples in the i^{th} class as positive examples and all the rest as negative examples. In the recognition phase, a test example is presented to all N SVMs and is labeled according to the maximum output among the N classifiers. The disadvantage of this method is that it is difficult to train as the number of training samples is large. Each of the N classifiers is trained using all available samples.

B. *One-against-one (OvO)*

This algorithm constructs $\frac{N(N-1)}{2}$ 2-class classifiers, using all the binary pair-wise combinations of the N classes. Each classifier is trained using the samples of the first class as positive examples and the samples of the second class as negative examples. To combine these classifiers, it naturally adopts Max Wins algorithm that finds the resultant class by first voting the classes according to the results of each classifier and then choosing the class that is voted most [7]. The disadvantage of this method is that every test sample has to be presented to large number of classifiers $\left(\frac{N(N-1)}{2}\right)$. This results in faster training but slower testing, especially when the number of the classes in the problem is big [8].

C. *Directed acyclic graph SVM (DAGSVM)*

Introduced by Platt [1] the algorithm for training a $\frac{N(N-1)}{2}$ classifiers is the same as in one-against-one. In the recognition phase, DAGSVM depends on a rooted binary directed acyclic graph to make a decision [9]. When a test sample reaches the leaf node, the final decision is made. A test example is presented only to the $N-1$ SVMs in the nodes on the decision path. This results in significantly faster testing while keeping similar recognition rate as One-against-one.

D. *Binary Tree of SVM (BTS)*

This method uses multiple SVMs arranged in a binary tree structure [10]. A SVM in each node of the tree is trained using two of the classes. The algorithm then employs probabilistic outputs to measure the

similarity between the remaining samples and the two classes used for training. All samples in the node are assigned to the two subnodes derived from the previously selected classes by similarity. This step repeats on every node until each node contains only one class samples. The main problem that should be considered seriously here is training time, because, one has to test all samples in every node to find out which classes should be assigned to which subnode while building the tree. This may decrease the training performance considerably for huge training datasets.

In this paper we propose a binary decision tree architecture that uses SVMs for making the binary decisions in the nodes. The proposed classifier architecture SVM-DTA (Support Vector Machines with Decision Tree Architecture), takes advantage of both the efficient computation of the tree architecture and the high classification accuracy of SVMs. Utilizing this architecture, $N-1$ SVMs are needed to be trained for an N class problem, but only at most $\lceil \log_2 N \rceil$ SVMs are required to be consulted to classify a sample. This can lead to a dramatic improvement in recognition speed when addressing problems with big number of classes.

IV. SUPPORT VECTOR MACHINES IN DECISION TREE ARCHITECTURE

As shown on Figure 3, the SVM-DTA solves an N-class pattern recognition problem utilizing a binary tree, in which each node makes binary decision using a SVM. The hierarchy of binary decision subtasks should be carefully designed before the training of each SVM classifier.

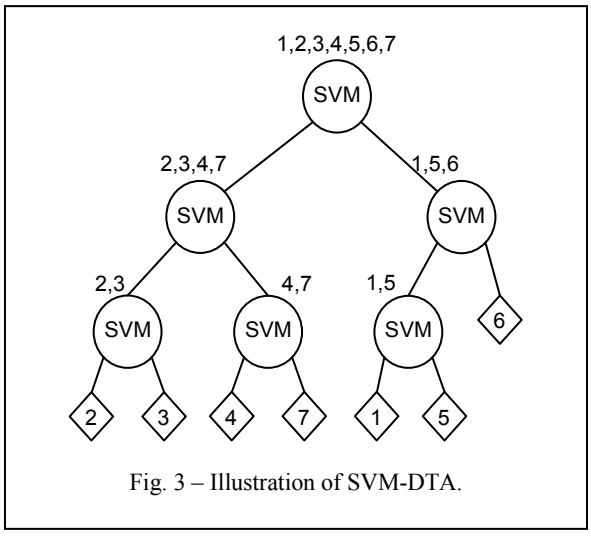

Fig. 3 – Illustration of SVM-DTA.

The recognition of each pattern 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 right sub-tree. Each of these groups may contain multiple classes. This is repeated downward the tree until the sample reaches a leaf node that represents the class it has been assigned to.

There exist many ways to divide the classes into 2 groups, and it is critical to have proper grouping for the good performance of SVM-DTA. For consistency between the clustering model which divided the classes into two groups in every node and the way SVM calculates the decision hyperplane, the clustering model utilizes distance measures at the kernel space, not at the input space. Because of this, all training samples are modified with the same kernel function that is to be used in the training phase. To measure the distance between two groups of classes we decided to use the Mahalanobis distance. It is also called quadratic distance. It measures the separation of two groups of samples. Suppose we have two groups with means $\hat{x}_i$ and $\hat{x}_j$, Mahalanobis distance is given by

$$d_{ij} = \left(\left(\hat{x}_i - \hat{x}_j \right)^T S^{-1} \left(\hat{x}_i - \hat{x}_j \right) \right)^{\frac{1}{2}}, \qquad (11)$$

where S^{-1} is an inverse pooled covariance matrix. This matrix is computed using weighted average of covariance matrices of both of the groups.

The SVM-DTA method that we propose is based on recursively dividing the classes in two disjoint groups in every node of the decision 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 a clustering algorithm according to their class membership and their interclass distance in kernel space.

Let's take a set of samples $x_1, x_2, ..., x_n$ labeled each one by $y_i \in \{c_1, c_2, ..., c_k\}$ where k is the number of classes. SVM-DTA method starts with dividing the classes in two disjoint groups g_1 and g_2. This is performed by calculating the gravity centers and the covariance matrices for the k different classes. Then, the two classes that have the biggest Mahalanobis distance from each other are assigned to each of the two clustering groups. After this, a pair of classes, each of which is closest to one of the two clustering groups are found and assigned to the corresponding group. The closeness of the classes is measured using the Mahalanobis distance to the corresponding group in kernel space. The group parameters (gravity center and covariance matrix) are then recalculated to represent the addition of the samples of the newly added class to the group, for both of the groups.

The process continues by finding the next pair of unassigned classes each of which is closest to one of the two clustering groups, assigning them to the corresponding group and updating the group's parameters. This is repeated until all classes are assigned to one of the two possible groups. In case

there is only one unassigned class left, it is assigned to the closer clustering group.

This defines a grouping of all the classes in two disjoint groups of classes. This grouping is than used to train a SVM classifier in the root node of the decision 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. This procedure leads to binary tree for the SVM-DTA that will always be as balanced as possible, resulting in best decision efficiency.

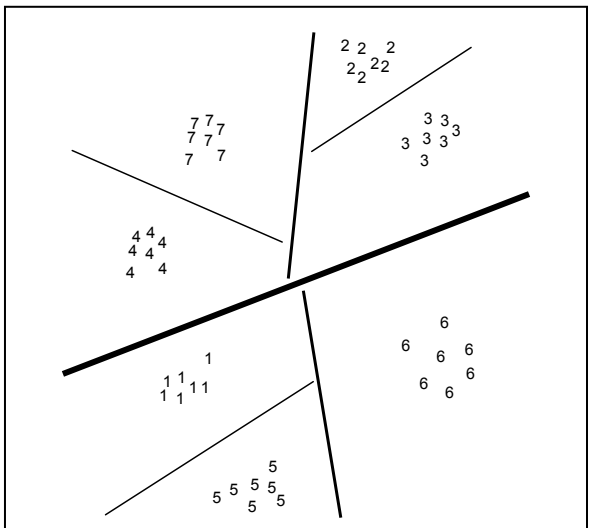

Fig. 4 – SVM-DTA divisions of the seven classes.

For example, Fig. 4 illustrates grouping of 7 classes while Fig. 3 shows the corresponding decision tree of SVMs. After calculating the class parameters, the groups c_2 and c_5 are found to be furthest apart considering their Mahalanobis distance and are assigned to group g_1 and g_2 correspondingly. Closest to the group g_1 is class c_3 and closest to the group g_2 is class c_1, so they are assigned to the corresponding groups and both group parameters are recalculated. In the next step c_7 is assigned to g_1 and c_6 is assigned to g_2. Finally c_4 is assigned to g_1. This completes the first round of the grouping that defines the classes that will be transferred to the left and the right subtree of the root node. The SVM classifier in the root is trained by considering samples from the classes $\{c_2,\ c_3,\ c_4,\ c_7\}$ as positive examples and samples from the classes $\{c_1,\ c_5,\ c_6\}$ as negative examples.

The grouping procedure is repeated independently for the classes of the left and the right subtree of the root which results in grouping c_7 and c_4 in $g_{1,1}$ and c_2 and c_3 in $g_{1,2}$ in the left node of the tree and c_1 and c_5 in $g_{2,1}$ and c_6 in $g_{2,2}$ in the right node of the tree. The concept is repeated for each SVM associated to a node in the taxonomy. This will result in training only $k-1$ SVMs for solving a k-class problem.

V. RELATED WORK AND DISCUSSION

A multistage SVM (MSVM) for multi-class problem was proposed in [11]. It uses Support Vector Clustering (SVC) [12] to divide the training data into two parts and then a binary SVM is trained. For each part, the same procedure recursively takes place until the binary SVM gives an exact label of class. The unsolved problem for MSVM is how to control the SVC to divide the training dataset into exact two parts. However, this procedure is painful and unfeasible, especially for large datasets. The training set from one class could lie in both clusters.

SVM-DTA is a feasible solution for multi-class SVM. Compared with the OvA, OvO, DAGSVM and BTS, SVM-DTA has its advantages.

The training time for a binary SVM is estimated empirically by a power law [13] to be $T \approx \alpha N^2$, where N is the number of training samples and α is some proportionality constant. Following this law, the estimated training time for OvA is

$$T_{OvA} \approx K\alpha N^2, \qquad (11)$$

where K is the number of classes in the problem.

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 $\frac{2N}{K}$ samples. Therefore, the training time for OvO is:

$$T_{OvO} \approx \alpha \frac{K(K-1)}{2}\left(\frac{2N}{K}\right)^2 \approx 2\alpha N^2, \qquad (12)$$

The training time for DAGSVM is same as OvO.

As for BTS and SVM-DTA, the training time is summed over all the nodes in the $\lceil \log_2(K) \rceil$ levels.

In the i^{th} level, there are 2^{i-1} nodes and each node uses $2\frac{N}{K}$ for BTS and $\frac{N}{2^{i-1}}$ for SVM-DTA training samples. Hence, the total training time for BTS is:

$$T_{BTS} \approx \sum_{i=1}^{\lceil \log_2(K) \rceil} \alpha 2^{i-1} \left(2\frac{N}{K}\right)^2 \approx \alpha \left(2\frac{N}{K}\right)^2 \sum_{i=1}^{2\lceil \log_2(K) \rceil} 2^{i-1} \approx 4\alpha \frac{N}{K}^2, \quad (13)$$

and for SVM-DTA is:

$$T_{SVM-DTA} \approx \sum_{i=1}^{\lceil \log_2(K) \rceil} \alpha 2^{i-1} \left(\frac{N}{2^{i-1}}\right)^2 \approx 2\alpha N^2 \quad (14)$$

Here, we have to notice that $T_{SVM-DTA}$ does not include the time to build the hierarchy structure of the K classes, since doing so will not consume much time and the quadratic optimization time dominates the total SVM training time. On the other hand, in the process of the building the tree, BTS requires testing of each trained SVM with all the training samples in order to determine the next step.

According the empirical estimation above, we can see that the training speed of SVM-DTA is comparable with OvA, OvO, DAGSVM and BTS.

In testing, DAGSVM is faster than OvO and OvA, since it requires only $K-1$ binary SVM evaluations. SVM-DTA is even faster than DAGSVM because the depth of the SVM-DTA decision tree is $\lceil \log_2 K \rceil$ at most, which is superior to $K-1$ especially when $K >> 2$.

While testing, for each sample the inner product between the feature vector of the sample and all the support vectors of the model are calculated. The total number of support vectors in the trained model contributes directly to the major part of the evaluation time, which was also confirmed by the experiments.

VI. EXPERIMENTS

In this section, we present the results of our experiments with two multi-class problems. The performance was measured on the problem of recognition of handwritten digits and letters.

Training and testing of the SVMs was performed using a custom developed application that uses the Torch library [14]. For solving the partial binary classification problems SVMs using Gaussian kernel were used.

Here, we compare the results of the proposed SVM-DTA method with the following methods:

1) one-against-all (OvA);
2) one-against-one (OvO);
3) DAGSVM;
4) BTS;

The most important criterion in evaluating the performance of a classifier is usually its recognition rate, but very often the training and testing time of the classifier are equally important.

In our experiments 2 different multi-class classification problems were addressed by each of the 5 previously mentioned methods. For every method the training and testing time and the recognition performance were recorded.

The first is a 10-class problem from the UCI Repository [15] of machine learning databases: pendigit. Pendigit has 16 features, 7494 training samples, and 3498 testing samples.

The second problem was recognition of isolated handwritten letters – a 26-class problem from the Statlog collection [16]. Statlog-letter contains 15.000 training samples, and 5.000 testing samples, while each sample is represented by 16 features.

The classifiers were trained using all training samples for the set and were evaluated by recognizing all the test samples for the corresponding set. All tests were performed on personal computer with Intel Core2Duo processor at 1.86GHz on Windows XP.

Table 1. Recognition results, training and testing times for the pendigit dataset

Classifier	Error rate (%)	Time (s)	
		Train	Test
OvA	1.70	4.99	1.75
OvO	1.94	3.11	3.63
DAGSVM	1.97	3.11	0.55
BTS	1.94	5.21	0.57
SVM-DTA	1.88	1.67	0.57

Table 2. Recognition results, tarining and testing times for the statlog dataset.

Classifier	Error rate (%)	Time (s)	
		Train	Test
OvA	3.20	554.2	119.5
OvO	4.72	80.9	160.5
DAGSVM	4.74	80.9	12.5
BTS	4.70	387.1	17.2
SVM-DTA	4.36	62.6	13.0

Table 1 and Table 2 show the results of the experiments using 5 different approaches on each of the 2 data sets. The first column of each of the tables denotes the multiclass SVM classification method: one-against-all (OvA), one-against-one (OvO), DAGSVM, BTS and SVM-DTA. In the second column the error-rate is given. The last two columns present the training time and the testing time for the corresponding method.

From the results in Table 1 we can see that methods one-against-one (OvO), DAGSVM, BTS and our method SVM-DTA can reach almost the same accuracy. The method one-against-all (OvA) is more accurate than the other methods. The time needed to train the 10 classifiers for the OvA approach took about 1.5 times longer than training

the 45 classifiers for the OvO and DAGSVM methods. SVM-DTA is the fastest one in the training phase. The DAGSVM method showed to be the fastest in the recognition phase but also produces the biggest error rate. Testing time is comparable in methods DAGSVM, BTS and SVM-DTA and they are noticeably better then testing time of one-against-all (OvA) and one-against-one (OvO) methods. However, if the number of the classes is relatively small, the advantage of SVM-DTA is not that evident.

The second problem was recognition of handwritten letters from the Statlog database [16]. Table 2 presents the results of the experiment for this 26-class problem. Again the OvA method showed the lowest error rate but the longest training time. The OvO, DAGSVM and the BTS method achieved very similar error rates that were about 1.5% higher than the OvA method. The DAGSVM is again fastest in recognition being almost 10 times faster than OvA. The time needed for training of the 26 one-against-all SVMs was almost 7 times longer than the time for training the 325 one-against-one SVMs. The BTS method showed the lowest error rate of the methods that use one-against-one SVMs. The SVM-DTA method showed better recognition rate than the methods using one-against-one SVMs while being only slightly slower in recognition than DAGSVM and the fastest while training.

VII. CONCLUSION

We have presented a novel method of arranging a binary classifiers like support vector machines in order to solve a multi-class problem. The proposed Support Vector Machines in Decision Tree Architecture (SVM-DTA) method was designed to provide superior recognition speed utilizing decision tree architecture, while keeping comparable recognition rate to the other known methods. Clustering algorithm that utilizes Mahalanobis distance measures at the kernel space is used to convert the multi-class problem into series of binary decision problems in a binary decision tree structure, in which the binary decisions are made by the SVMs. The experiments performed on 2 different datasets of handwritten digits and letters have shown that this method has one of the fastest training times and comparable testing times while keeping similar recognition rate to the other methods. SVM-DTA also shows very good scalability with the number of the classes in the classification problem as its complexity grows much slower than the other methods, making it a preferable choice for multiclass classification problems with large number of classes.

REFERENCES

[1]. V. Vapnik. The Nature of Statistical Learning Theory, second ed.. Springer, New York, 1999.

[2]. C. J. C. Burges. A tutorial on support vector machine for pattern recognition. Data Min. Knowl. Disc. 2 (1998) 121.

[3]. T. Joachims. Making large scale SVM learning practical. in B. Scholkopf, C. Bruges and A. Smola (eds). Advances in kernel methods-support vector learning, MIT Press, Cambridge, MA, 1998.

[4]. R. Fletcher. Practical Methods of Optimization. 2nd Ed. John Wiley & Sons. Chichester (1987).

[5]. J. Weston, C. Watkins. Multi-class support vector machines. Proceedings of ESANN99, M. Verleysen, Ed., Brussels, Belgium, 1999.

[6]. V. Vapnik. Statistical Learning Theory. Wiley, New York, 1998.

[7]. J. H. Friedman. Another approach to polychotomous classification. Technical report. Department of Statistics, Stanford University, 1997.

[8]. P. Xu, A. K. Chan. Support vector machine for multi-class signal classification with unbalanced samples. Proceedings of the International Joint Conference on Neural Networks 2003. Portland, pp.1116-1119, 2003.

[9]. Platt, N. Cristianini, J. Shawe-Taylor. Large margin DAGSVM's for multiclass classification. Advances in Neural Information Processing System. Vol. 12, pp. 547–553, 2000.

[10]. 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.

[11]. X. Liu, H. Xing, X. Wang. A multistage support vector machine. In The 2nd International Conference on Machine Learning and Cybernetics, pages 1305-1308, 2003.

[12]. A. Ben-Hur, D. Horn, H. Siegelmann, V. Vapnik. Support vector clustering. Journal of Machine Learning Research, vol. 2:125-137, 2001.

[13]. J. Platt. Fast training of support vector machines using sequential minimal optimization. In Advances in Kernel Methods - Support Vector Learning, pages 185-208, Cambridge, MA, 1999. MIT Press.

[14]. R. Collobert, S. Bengio, J. Mariéthoz. Torch: a modular machine learning software library. Technical Report IDIAP-RR 02-46, IDIAP, 2002.

[15]. C. Blake, E. Keogh and C. Merz. UCI Repository of Machine Learning Databases, (1998). Statlog Data Set, http://archive.ics.uci.edu/ml/datasets.html [Online]

[16]. Statlog Data Set, http://archive.ics.uci.edu/ml/datasets/Letter+Recognition [Online]

Gjorgji Madzarov received his BSc in computer science, automation and electrical engineering in 2007 from the Faculty of Electrical Engineering and Information Technologies, University "St. Cyril and Methodius" in Skopje. Now he is working on his MSc thesis in the area of machine-learning and artificial intelligence. His employment experience includes NETCETERA-Skopje. Now he is a teaching and research assistant at the Faculty of Electrical Engineering and Information Technologies in Skopje, Macedonia.

Dejan Gjorgjevikj received his BSc, in electrical engineering, and his MSc and PhD in computer science and engineering from the Faculty of Electrical Engineering, University "St. Cyril and Methodius" - Skopje, in 1992, 1997 and 2004, respectively. He is currently a professor at the Faculty of Electrical Engineering and Information Technologies, University "St. Cyril and Methodius" in Skopje, Macedonia. His research interests include artificial

intelligence, machine learning, computer vision, pattern recognition and software engineering. He is a member of IEEE and ACM.