Distances Tree as SVM Ensemble in Digits Recognition Task

Marcin Luckner[1]

[1]Faculty of Geodesy and Cartography,
Warsaw University of Technology,
Plac Politechniki 1, 00-660 Warsaw, Poland.
email: mluckner@gik.pw.edu.pl

Abstract

This paper presents several algorithms that creates a classification system based on SVM classifiers grouped in a tree structure. Analysis of similarity between classes allows to reduce of number of used SVM in comparison to DAGSVM method without major reduction of an accuracy. Practical tests of map texts recognition task are also presented.

Keywords: Pattern Recognition, SVM, decisions tree

1. Introduction

The SVM ensemble successful resolves classification tasks. However a brief review of them presented in Section 2 shows that their structure not depends on a data distribution.

Distances trees are proposed as an alternative to knows algorithm. In this method a similarity between classes is considered in creation of ensemble structure. Three different algorithm are proposed in Section 3.2. All of them are tested in digits recognition task and results are presented in Section 4

2. Multiclass SVM ensembles

A single SVM classifier [1] splits a data space binary. To resolve a multiclass classification task, an ensemble of classifiers should be used. A review of method for multiclass Support Vector Machines is presented in [2].

The first group are methods that resolve a classification task as a single optimization problem [1, 3]. Unfortunately, a computation cost for those "all–together" methods is too expensive for a practical use. For that reason "all–together" strategy will be not discussed in this paper.

The second group of methods use "one–against–all" [4] strategy where each class is represented by a single SVM classifier. The classifier decides that example is a member of analyzed class or not. If more that one class has a positive response a distance of a point that represents the example to a hyperplane defined by the SVM classifier is taken into consideration. The method needs k classifiers to recognize k classes.

A cost of the method can be reduced if is implemented as a tree structure of SVMs where each classifier splits data space between a single class and the rest of classes not separated on higher levels. An application of the tree is presented in [5]. The tree needs $k-1$ classifiers to recognize k classes. What more important, because a number of separated classes is decreased on each level of tree, also a number of supported vectors will be limited.

The last group of method relies on "one–against–one" [6] strategy. Each class is tested against each. The basic algorithm counts wins for each class to select a winner. This method needs $\frac{1}{2}k^2-k$ classifiers to recognize k classes. The number of classifiers is significant larger from used in "one–against–all" method. However, a splits are done between single classes and a number of supported vectors should be reduced.

Methods based on trees and graphs have been proposed to reduce a number of used classifiers. In the paper [7] the strategy is implemented as a bottom–up tournament tree. Random paired classes described in leaves are confronted and the winer goes up in the tree. The class from the root is a classification result.

More popular is an algorithm based on Directed Acyclic Graph, DAGSVM [8]. The graph is top–down oriented. From a random ordered classes the first and the second one are confronted. On a lower level the winner is presented against a next class from the list. Finally, a class that defeats all opponents is defined in a leaf of the graph.

Both methods needs to create $\frac{1}{2}k^2-k$ classifiers for k classes. However, in a decision path only $k-1$ classifiers is used. An interesting innovation is presented in [9] where a distance between classes is used to reduce a number of data splits. For each class the rest of classes is ordered by distance to considered class. A split between the class and the first class in line is created. For next classes a split is calculated if and only if any of created hyperplane cannot be used.

All presented algorithms [5, 7, 8] use only a number of recognized classes to determine a tree or graph structure. Only in [9] also a distribution of classes in a data space is considered to improve a classification system. For that reason a new solution based on a similarity between classes is proposed.

3. Distance SVM trees

Creation of distances SVM tree processes in three steps. First, a distance matrix is calculated for all classes. For that a metric has to be chosen and a distance between classes defined. Those selections determine a structure of a created tree. Each node in the tree is connected with a SVM classifier that splits a data space between examples determined by leaves those are the node successors.

3.1. Classes similarity

As a measure of similarity between classes a distance can be used. For a small distance a similarity is high, for a large distances similarity is low. A distance between examples in classes is defined by used metric, a distance between classes may C_X and C_Y be defined in several ways for example as a distance between nearest elements:

$$d(C_X, C_Y) = \min_{x\in C_X y\in C_Y} d(x,y),$$

between farthest elements:

$$d(C_X, C_Y) = \max_{x\in C_X y\in C_Y} d(x,y),$$

the average from distances:

$$d(C_X, C_Y) = \frac{\sum_{x\in C_X y\in C_Y} d(x,y)}{n_{C_X} n_{C_Y}},$$

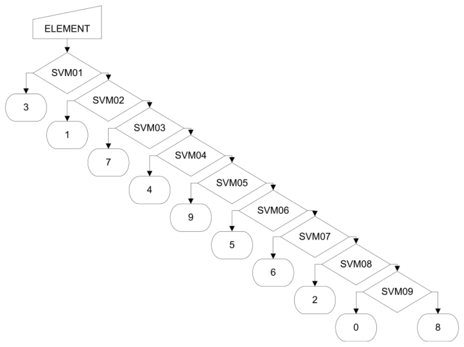

Fig. 1: One–against–all tree

the distance between centroids:

$$d(C_X, C_Y) = d\left(\frac{\sum_{x \in C_X} x}{n_{C_X}}, \frac{\sum_{y \in C_Y} y}{n_{C_Y}}\right).$$

Where n_{C_X} and n_{C_Y} are numbers of elements in classes C_X and C_Y. The function $d(x, y)$ is distance in chosen metric calculated for elements of classes C_X i C_Y.

For large data sets the last definition should be used to avoid calculation of a huge distance matrix.

In described test an Euclidean metric has been used. Tests for different metrics are presented in [10].

3.2. Tree structure creation algorithms

In this section various algorithm that creates tree structure are presented. All base on distances between classes.

3.2.1. One–against–all tree

This algorithm creates a degenerated tree. Each node has at least one leaf as a child. The tree is created by Algorithm 1.

Distances between classes have not influence into the tree structure. Always a degenerated tree is created. However, an order of recognized classes is changing.

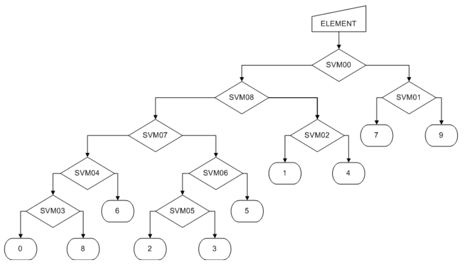

Fig. 2: Grouping tree

```
Data:
$S_c$ - set of recognized classes,
$M_d(S_c)$ - distances matrix for $S_c$
while $c(S_c) > 1$ do
    Find a class $C_m$ with the
    maximum sum of distances to
    the rest of classes;
    Create a SVM that splits data
    space between $C_m$ and $S_c \setminus C_m$;
    $S_c = S_c \setminus C_m$;
end
```

Algorithm 1: One-against-all tree

3.2.2. Distance tree

A structure of distance tree bases on classes similarity. In each step nearest classes are grouped and a SVM classifier that splits them is created. Next, classes are replaced by their union and the algorithm starts over. The tree is created by Algorithm 2.

```
Data:
$S_c$ - set of recognized classes,
$M_d(S_c)$ - distances matrix for $S_c$
while $c(S_c) > 1$ do
    Find nearest classes
    $C_x, C_y \in S_c$;
    Create a SVM that splits data
    space between $C_x$ and $C_y$;
    $S_c = S_c \setminus C_x$;
    $S_c = S_c \setminus C_y$;
    $S_c = S_c \cup \{C_x \cup C_y\}$;
    Update $M_d$;
end
```

Algorithm 2: Distance tree

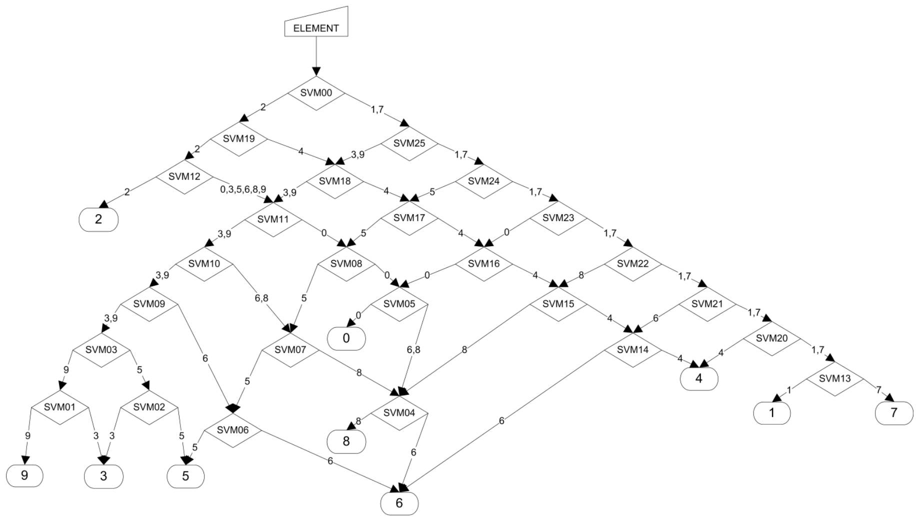

Fig. 3: Grouping graph

Because, a structure depends on a distances matrix for different metrics different trees will be created.

3.2.3. Distance graph

In the tree created in section 3.2.2 for each class only the nearest class is selected to create a split by a SVM classifier. It is efficient strategy for recognition tasks with limited number of similar classes. However, such structure cannot be able to lift at least tree very similar classes. Only for two of them a split will be implemented directly. For that reason a modified algorithm is proposed.

Grouped classes will be no longer rejected from a classes set. Already used classes can be included into next groups if level of similarity is essential. To avoid of cycles a class cannot be connected with their subset. Because, a size of the classes set will be not longer decremented a new stop condition has to be implemented. The modified algorithm is described in 3.

```
Data:
$S_c$ - set of recognized classes,
$M_d(S_c)$ - distances matrix for $S_c$
repeat
    Find nearest classes
    $C_x, C_y \in S_c$ but
    $C_x \nsubseteq C_y \wedge C_y \nsubseteq C_x$;
    Create a SVM that splits data
    space between $C_x$ and $C_y$;
    $S_c = S_c \cup \{C_x \cup C_y\}$;
    Update $M_d(S_c)$ ;
until $C_x, C_y$ found ;
```

Algorithm 3: Distance graph

In such tree nodes can be multiplied. Each duplicated node duplicates its subtree. For that reason all instances can be treat as a single instance with more that one path from the root. After such modification a tree becomes a graph.

3.3. Learning SVM in node

Each node of tree is bound up with a SVM classifier. If successors of the node are leaves with connected classes the classifiers split a data space be-

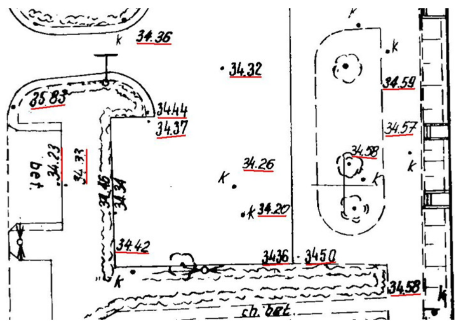

Fig. 4: Classification task: digs from cadastral map

	Method			
	OAO	OAA	Tree	Graph
Avg	95.75	93.35	93.02	95.25
Min	82.76	78.43	63.22	83.91
Max	97.94	97.67	97.01	97.76
Sdv	4.95	5.91	9.92	4.08
Svm	40	9	9	26

Table 1: Comparison of classification methods.

tween them. In other case for both subtrees of node separated groups are created from all leaves that are members of subtrees. The split is calculated between those groups.

If a structure is represented by a graph there is a possibility that a leaf is a member of both subtrees. In such case a class connected with the leaf is eliminated from both groups.

4. Tests

In discussed classification task one class from *0* to *9* should be assigned to a features vector. Recognized elements are taken from cadastral maps such as presented in Figure 4. For recognized elements translation, scale and rotation are normalized [11]. Next, low cost features such as histograms and projection are calculated.

From five sheets of cadastral maps 7081 elements have been extracted. 75 percent of them has been used as a learning probe, the rest as a testing probe. All presented results are obtained for tested probe.

The classification task has been resolved by tree structures described in section 3.2. For comparison also "one–against-one" algorithm has been used.

For all SVM classifiers a linear kernel has been used. A parameter C [12] that determines penalization for elements from learning set that are bad classified has been arbitrary established to 10.

For the method "one–against-one" 40 classifiers has been created for 10 recognized classes. An accuracy was 95.75 percent.

A tree that implements a strategy "one–against-all" has been created with 9 classifiers. The order of classifiers in the tree structure depends on the average distance between classes calculated in Euclidean metric. The tree presented in Figure 1 has 93.35 percent accuracy.

The tree based on the algorithm presented in Section 3.2.2 also is based on 9 classifiers but its structure is not degenerated. The average distance between classes calculated in Euclidean metric was also used in this case. The accuracy is only 93.02 percent. An analysis of a misclassification matrix shows that the main origin of errors is in elements of the class *9* incorrectly recognized as a class *3*. As is shown in Figure 2 those classes are not split directly by SVM classifiers but as members of groups.

For a graph created by the algorithm described in Section 3.2.3 a number of classifiers depends not only on a number of classes but also on a similarity of classes. For recognized classes

26 classifiers have been created. It is much more that for trees algorithms but still considerably less that for "one–against–one" strategy. A structure of the graph, presented in Figure 3 depends on distances between centroids calculated in Euclidean metric. An accuracy is 95.25 percent which is nearly as good as "one–against–one" method. The misclassification between classes *9* and *3* has been reduced by the SVM01 classifier that splits ones directly.

5. Conclusions

As is shown in Table 1 distance graph results are similar to "one–against–one" method but with a significant reduced number of used SVM classifiers. In the future distance trees should be compared with different methods that takes distances into consideration in the decision process [9].

References

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

[2] C. W. Hsu and C. J. Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on neural networks*, 13(2):415–425, 2002.

[3] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *COLT 2000 Proceedings*, pages 35–46, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[4] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: a case study in handwritten digit recognition. *Pattern Recognition*, 2:77–82 o.2, Oct 1994.

[5] S. Han, W. You, and H. Li. Application of binary tree multi-class classification algorithm based on svm in shift decision for engineering vehicle. *ICCA 2007 Proceedings*, pages 1833–1836, 2007.

[6] U. H.-G. Kressel. *Pairwise classification and support vector machines*, pages 255–268. MIT Press, Cambridge, MA, USA, 1999.

[7] G. Guo, S. Z. Li, and K. L. Chan. Support vector machines for face recognition. *Image and Vision Computing*, 19(9-10):631–638, August 2001.

[8] J. Platt, N. Cristianini, and J. ShaweTaylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems 12*, pages 547–553, 2000.

[9] R. Debnath, N. Takahide, and H. Takahashi. A decision based one-against-one method for multiclass support vector machine. *Pattern Analysis and Applications*, 7(2):164–175, July 2004.

[10] M. Luckner. *Comparison of Hierarchical SVM Structures in Letters Recognition Task*, pages 291–302. Computational Intelligence: Methods and Applications. Academic Publishing House EXIT, Warsaw, 2008.

[11] M. P. Deseilligny, H. Le Men, and G. Stamon. Character string recognition on maps, a rotation-invariant recognition method. *Pattern Recognition Letters*, 16(12):1297–1310, December 1995.

[12] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273– 297, September 1995.