Chapter 2

SVM Classifiers – Concepts and Applications to Character Recognition

Antonio Carlos Gay Thomé

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/52009

1. Introduction

Support Vector Machines – SVMs, represent the cutting edge of ranking algorithms and have been receiving special attention from the international scientific community. Many successful applications, based on SVMs, can be found in different domains of knowledge, such as in text categorization, digital image analysis, character recognition and bioinformatics.

SVMs are relatively new approach compared to other supervised classification techniques, they are based on statistical learning theory developed by the Russian scientist Vladimir Naumovich Vapnik back in 1962 and since then, his original ideas have been perfected by a series of new techniques and algorithms.

Since the introduction of the concepts by Vladimir, a large and increasing number of researchers have worked on the algorithmic and the theoretical analysis of SVM, merging concepts from disciplines as distant as statistics, functional analysis, optimization, and machine learning. The soft margin classifier was introduced few years later by Cortes and Vapnik [1], and in 1995 the algorithm was extended to the regression case.

There are several published studies that compare the paradigm of neural networks against to the support vector machines. The main difference between the two paradigms lies in how the decision boundaries between classes are defined. While the neural network algorithms seek to minimize the error between the desired output and the generated by the network, the training of an SVM seeks to maximize the margins between the borders of both classes.

SVM approach has some advantages compared to others classifiers. They are robust, accurate and very effective even in cases where the number of training samples is small. SVM technique also shows greater ability to generalize and greater likelihood of generating good classifiers.

By nature SVMs are essentially binary classifiers, however, based on several researchers' contributions they were adapted to handle multiple classes cases. The two most common approaches used are the One-Against-All and One-Against-One techniques, but this scenario is still an ongoing research topic.

In this chapter we briefly discuss some basic concepts on SVM, describe novel approaches proposed in the literature and discuss some experimental tests applied to character recognition. The chapter is divided into 4 sections. Section 2 presents the theoretical aspects of the Support Vector Machines. Section 3 reviews some strategies to deal with multiple classes. Section 4 details some experiments on the usage of One-Against-All and One-Against-One approach applied to character recognition.

2. Theoretical foundations of the SVM

Support vector machines are computational algorithms that construct a hyperplane or a set of hyperplanes in a high or infinite dimensional space. SVMs can be used for classification, regression, or other tasks. Intuitively, a separation between two linearly separable classes is achieved by any hyperplane that provides no misclassification on all data points of any of the considered classes, that is, all points belonging to class A are labeled as +1, for example, and all points belonging to class B are labeled as -1.

This approach is called linear classification however there are many hyperplanes that might classify the same set of data as can be seen in the figure 1 below. SVM is an approach where the objective is to find the best separation hyperplane, that is, the hyperplane that provides the highest margin distance between the nearest points of the two classes (called functional margin). This approach, in general, guarantees that the larger the margin is the lower is the generalization error of the classifier.

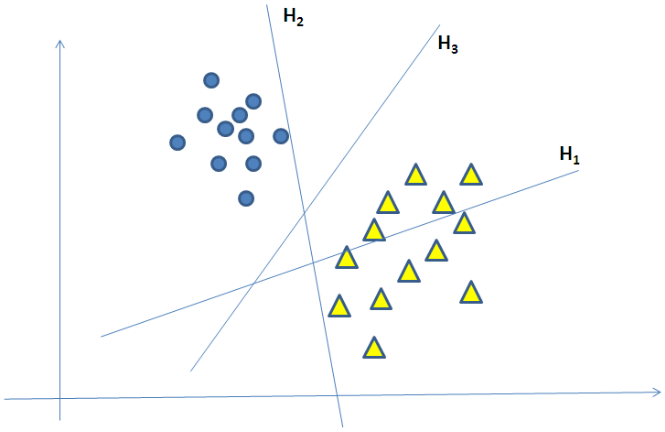


Figure 1. Separation hyperplanes. H_1 does not separate the two classes; H_2 separates but with a very tinny margin between the classes and H_3 separates the two classes with much better margin than H_2

If such hyperplane exists, it is clear that it provides the best separation border between the two classes and it is known as the maximum-margin hyperplane and such a linear classifier is known as the maximum margin classifier.

2.1. Brief history

Research on pattern recognition started in 1936 through the work done by R. A. Fisher who suggested the first algorithm for pattern recognition [2]. After him we have the work done by Frank Rosemblat in 1957 that invented the nowadays well known linear classifier named PERCEPTRON that is the simplest kind of feed forward neural network [3]. In 1963 Vapnik and Lerner introduced the Generalized Portrait algorithm (the algorithm implemented by support vector machines is a nonlinear generalization of the Generalized Portrait algorithm) [4]. Aizerman, Braverman and Rozonoer in 1964, introduced the geometrical interpretation of the kernels as inner products in a feature space [5] and Cover in 1965 discussed large margin hyperplanes in the input space and also sparseness [6].

The field of statistical learning theory was first developed and proposed by Vapnik and Chervonenkis in 1974 [7] and, based on this theory, appears in the year of 1979, the first concepts about SVMs [8]. SVMs close to their current form were first introduced by Boser at al. with a paper presented at the COLT 1992 conference in 1992 [9].

2.2. Formal definition of the SVM classifier – The linear model

The surface model used by SVM to perform the separation is the hyperplane. Let then **W** and **b** be, respectively, the vector normal to the hyperplane and its displacement relative to the origin [10]. Thus, we have that the decision function for an input **x** is given by equation (1).

$$D(x) = W \bullet x - b \tag{1}$$

where,

$$x \in \begin{cases} A & if\ D(x) > 0 \\ B & if\ D(x) < 0 \end{cases} \tag{2}$$

As can be seen in figure 2 below, the distance from **x** (with signal) to the hyperplane is given by 3.

$$\frac{D(x)}{\|W\|} \tag{3}$$

Thus, D(**x1**) and D(**x2**) will have opposite signs (belong to different sets) if and only if **x1** and **x2** are on opposite sides of the separation hyperplane.

Figure 2 shoes that the Vector **W** is perpendicular to the hyperplane and the parameter $\frac{|b|}{\|W\|}$ determines the offset of the hyperplane from the origin along the normal vector. It is desired to choose **W** and **b** to maximize the margin M that represents the distance between the parallel hyperplanes that are as far apart as possible while still separating the both set of data. These two hyperplanes can be described respectively by the following equations (4).

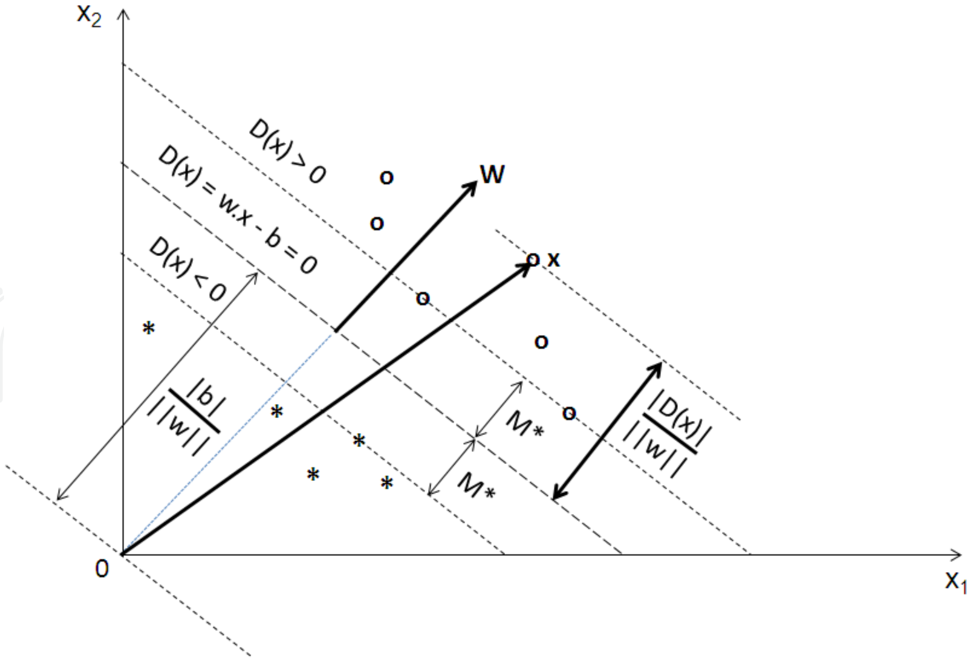


Figure 2. Example of the separating hyperplane (in two dimensions), distances and margins (from Boser et al, 1992 [9]).

$$\begin{aligned} &W \bullet x - b = +1 \\ &and \\ &W \bullet x - b = -1 \end{aligned} \tag{4}$$

Let the set of sample points be represented by $\mathbf{x}_1$, …, $\mathbf{x}_p$ and their respective group classification be represented by y_1, …, y_p where

$$y_i = \begin{cases} +1 & if\ x_i \in A \\ -1 & if\ x_i \in B \end{cases} \tag{5}$$

If the two groups of samples in the training data are linearly separable it is then possible to select the two hyperplanes in a way that there are no points between them and then try to maximize the distance between the two hyperplanes [11].

The distance between these two hyperplanes is given by $\frac{2}{\|W\|}$ and to maximize it implies to minimize **W** and, in order to prevent data points falling into the margin M, we add the following constraint to each equation (6):

$$\begin{aligned} &W \bullet x_i - b \geq +1 \quad \forall i, y_i = +1 \\ &and \\ &W \bullet x_i - b \leq -1 \quad \forall i, y_i = -1 \end{aligned} \tag{6}$$

Multiplying each equation by its corresponding y_i they are transformed into just one equation as following (7):

$$y_i \bullet (W \bullet x_i - b) \geq 1 \quad \forall i, i = 1 \dots p \tag{7}$$

Dividing now both sides of the equation by $\|w\|$ it turns into (8)

$$\frac{y_i \bullet (W \bullet x_i - b)}{\|w\|} \geq \frac{1}{\|w\|} = M \quad \forall i, i = 1 \dots p \tag{8}$$

To maximize M we need to minimize $\|w\|$ subject to the following constraint (9).

$$\begin{aligned} &\min\,(in\ w,b) \\ &\quad \|w\| \\ &subject\ to\ \forall i,\ i = 1 \dots p \\ &\quad y_i \bullet \left(w \bullet x_i - b\right) - 1 \geq 0 \end{aligned} \tag{9}$$

The optimization problem above is difficult to solve because it depends on $\|w\|$, the norm of **w**, which involves a square root. Fortunately it is possible to alter the equation substituting $\|w\|$ by $\frac{1}{2}\|w\|^2$ without changing the solution (the minimum of the original and the modified equations have the same **w*** and **b***). The problem now belongs to the quadratic programming (QP) optimization that is easier to be computed and is stated as in (10).

$$\begin{aligned} &\min\,(in\ w,b) \\ &\quad \frac{1}{2}\|w\|^2 \\ &subject\ to\ \forall i,\ i = 1 \dots p \\ &\quad y_i \bullet \left(w \bullet x_i - b\right) - 1 \geq 0 \end{aligned} \tag{10}$$

The factor of 1/2 is used for mathematical convenience and the problem can now be solved by standard quadratic programming techniques. Applying non negative Lagrange multipliers α_i (i = 1 … p) to the objective function turns the problem into its dual form as in (11).

$$\begin{aligned} &L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{p} \alpha_i \left(y_i \left(w \bullet x_i - b \right) - 1 \right) \\ &subject\ to \\ &\quad \alpha_i \geq 0 \quad \forall i,\ i = 1 \dots p \end{aligned} \tag{11}$$

Considering now that in the solution point the gradient of **L()** is null, the equation can be handled in order to obtain a new quadratic programming problem as in (12):

$$\begin{aligned} &\left.\frac{\partial L}{\partial w}\right|_{w=w^*} = w^* - \sum_{i=1}^{p} \alpha_i y_i x_i = 0 \quad \therefore w^* = \sum_{i=1}^{p} \alpha_i y_i x_i \\ &\left.\frac{\partial L}{\partial w}\right|_{b=b^*} = -\sum_{i=1}^{p} \alpha_i y_i = 0 \quad \therefore \sum_{i=1}^{p} \alpha_i y_i = 0 \end{aligned} \tag{12}$$

In this case, the minimum point with respect to **w** and **b** is the same to the maximum with respect to $\boldsymbol{\alpha}$, and the problem can be stated as in (13).

$$\begin{aligned}
&\max\ (in\ \alpha_i) \\
&\quad \alpha^T \bullet 1 - \frac{1}{2}\alpha^T \bullet H \bullet \alpha \\
&subject\ to\ \forall i,\ i = 1 \ldots p \\
&\quad \alpha_i \geq 0, \\
&\quad \alpha^T \bullet y = 0
\end{aligned} \tag{13}$$

Where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_p)^T$, $\mathbf{y} = (y_1, \ldots, y_p)^T$, **0** and **1** have size p, and $H_{p \times p}$ is such that

$$H_{i,j} = y_i y_j x_i^T \bullet x_j \tag{14}$$

A condition imposed by the Kühn-Tucker Theorem is that

$$\alpha_i^* \left(y_i \left(w^* \bullet x_i - b^* \right) - 1 \right) = 0 \quad \forall i,\ i = 1 \ldots p \tag{15}$$

so that, if $\alpha_i^* \neq 0$ then

$$y_i \left(w^* \bullet x_i - b^* \right) - 1 = 0 \quad \forall i,\ i = 1 \ldots p \tag{16}$$

that is,

$$y_i \left(w^* \bullet x_i - b^* \right) = 1 \tag{17}$$

Any $\mathbf{x}_i$ that satisfies equation (17) is called **support vector** and the SVM trainings are reduced to the set of such vectors.

In the cases where the samples are not linearly separable the approach described above would diverge and grow arbitrarily. In order to deal with the problem it is then introduced a set of slack variables (δ) in equation (6) as showed in (18).

$$\begin{aligned}
&\quad D(x_i) = W \bullet x_i - b \geq +1 - \delta_i \quad \forall i,\ y_i = +1 \\
&and \\
&\quad D(x_i) = W \bullet x_i - b \leq -1 + \delta_i \quad \forall i,\ y_i = -1 \\
&where \\
&\quad \delta_i \geq 0, \quad \forall i,\ i = 1 \ldots p
\end{aligned} \tag{18}$$

These equations can be rewritten as

$$y_i D(x_i) \geq +1 - \delta_i \quad \forall i,\ i = 1 \ldots p \tag{19}$$

The slack variables provide some freedom to the system allowing some samples do not respect the original equations. It is necessary however to minimize the number of such samples and also the absolute value of the slack variables. The way to do this is introducing a penalization term into the objective function as follows (19):

$$\begin{aligned}&\max\ (in\ w, b)\\ &\quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{p}\delta_i\\ &subject\ to\ \forall i,\ i=1 \ldots p\\ &\quad y_i\left(w \bullet x_i - b\right) - 1 + \delta_i \geq 0,\\ &\quad \delta_i \geq 0\end{aligned} \tag{20}$$

Variable C indicates the strength of the penalization to be applied. Introducing Lagrange multipliers on the penalization variables the dual form of the problem becomes as in (21).

$$\begin{aligned}&L(w,b,\alpha,\delta,\lambda) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{p}\delta_i - \sum_{i=1}^{p}\alpha_i\left(y_i\left(w \bullet x_i - b\right) - 1\right) - \sum_{i=1}^{p}\lambda_i\delta_i\\ &subject\ to\ \forall i,\ i=1 \ldots p\\ &\quad \alpha_i \geq 0\\ &\quad \lambda_i \geq 0\end{aligned} \tag{21}$$

Where $\delta = \left(\delta_i \ldots \delta_p\right)^T\ and\ \lambda = \left(\lambda_i \ldots \lambda_p\right)^T$

From here, as before, the problem can be represented into its quadratic form in terms of α(22).

$$\begin{aligned}&\max\ (in\ \alpha_i)\\ &\quad \alpha^T \bullet 1 - \frac{1}{2}\alpha^T \bullet H \bullet \alpha\\ &subject\ to\ \forall i,\ i=1 \ldots p\\ &\quad \alpha^T \bullet y = 0,\\ &\quad 0 \leq \alpha \leq c\end{aligned} \tag{22}$$

Where c = (C … C) is a p dimension vector with all values equal C.

2.3. The non-linear model

Whereas the original problem as proposed by Vladimir Vapnik in 1979 [8], was stated for a finite dimensional space, it often happens that the sets to be discriminated are not linearly separable in their original space. For this reason, it was proposed by Isabelle Guyon, Bernhard Boser and Vapnik in 1992 [9], that the original finite-dimensional space was mapped into a higher-dimensional space, presumably making the separation easier in the new space.

In order to achieve non-linear separation, instead of generating a new quadratic programming problem as in previous section, it is possible to modify the vectors of the input space into vectors of a feature space through a chosen transform function ϕ◎ with N ≥ n and then compute the separation hyperplane on the feature space. Figure 3 shows an example of such scheme.

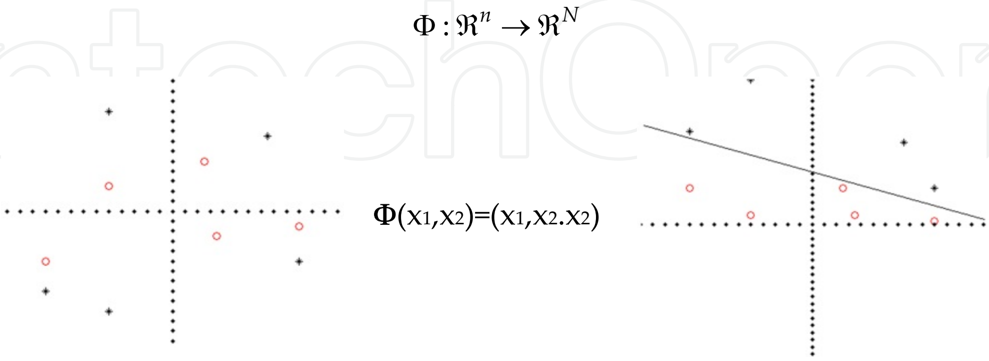


Figure 3. The transform function maintains the same dimension of the input space but makes the representation in feature space be linearly separable

The computation of the separation hyperplane is not done explicit on the feature space but using a scheme where every occurrence of **ϕ(u).ϕ(v)** is replaced by a function **K(u,v)** called kernel function and the H() function as seen before becomes (23):

$$H_{i,j} = y_i y_j K(x_i , x_j) \tag{23}$$

The optimum **W** vector is given by

$$w^* = \sum_{i=1}^{p} \alpha_i y_i \Phi(x_i) \tag{24}$$

And the support vector machine decision function becomes

$$D(x) = \sum_{i=1}^{p} \alpha_i y_i K(x_i, x) - b \tag{25}$$

To keep the computational load reasonable, the mapping used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function K(x,y) selected to suit the problem. The hyperplanes in the higher dimensional space are defined as the set of points whose inner product with a vector in that space is constant. The vectors defining the hyperplanes can be chosen to be linear combinations of feature vectors that occur in the data base. With this choice of a hyperplane, the points **x** in the feature space that are mapped into the hyperplane are defined by the relation:

$$\sum_{i} \alpha_i K(x_i, x) = constant \tag{26}$$

This approach allows the algorithm to find the maximum-margin hyperplane into the transformed feature space. The transformation may be non-linear and / or the transformed space may be of high dimension. The classifier, in the feature space, draws a hyperplane that represents a non-linear separation curve in the original input space.

If the kernel used is a Gaussian radial basis function, the corresponding feature space is a Hilbert space of infinite dimension. Maximum margin classifiers are well regularized, so the infinite dimension does not spoil the results. Some common kernels include:

Polynomial (homogeneous): $K(x_i, x) = (x_i \bullet x)^d$

Radial Basis Function: $K(x_i, x) = \exp(-\lambda \|x_i - x\|^2);\ \lambda > 0$

Gaussian Radial basis function: $K(x_i, x) = \exp(-\frac{\|x_i - x\|^2}{2\sigma^2})$

Sigmoid: $K(x_i, x) = \tanh(kx_i \bullet x + c);\ \textit{for some (but not every)}\ k > 0\ \textit{and}\ c < 0$

3. The multiclass classification strategies

Multiclass SVM approach aims to assign labels to a finite set of several elements based on a set of linear or non-linear basic SVMs. The dominant approach for doing so in the literature is to reduce the single multiclass problem into multiple binary problems [12 – 15].

Doing so, each of the problems can be seen then as a binary classification, which is assumed to produce an output function that gives relatively large values for those examples that belong to the positive class and relatively small values for the examples that belong to the negative class.

Two common methods to build such binary classifiers are those where each classifier is trained to distinguish: (i) one of the labels against to all the rest of labels (known as one-versus-all) [16] or (ii) every pair of classes (known as one-versus-one). Classification of new instances for one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest output function assigns the class. The classification of one-versus-one case is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally, the class with more votes determines the instance classification.

3.1. The one-versus-all strategy

One-Against-All multiclassifier is compound by a number of binary classifiers, one for each class. Using a Winner-Takes-All strategy, each binary classifier is trained taking the examples from one of the classes as positive and the examples from all other classes as negative. The multiclassifier output is activated for the class whose binary classifier gives the greatest output amongst all. Formally, given a vector **y** with the outputs of the binary classifiers, the multiclassifier generates a vector **L** = (l_1, . . . , l_s), in the following way (27):

$$i^* = \underset{i=1 \dots s}{\arg\max}\{y_i\},$$
$$L_i = \begin{cases} +1 & if\ i = i^*, i = 1 \dots s \\ -1 & otherwise \end{cases} \tag{27}$$

Where ‘s’ represents the total number of classes.

As seen, a one-against-all multiclassifier for 's' different classes requires the construction of 's' distinct binary classifiers, each one responsible for distinguishing one class from all the others. However, doing so does not guarantee that the resulting multi-class classifier is good. The problem is that all binary classifiers are assumed to show equal competence distinguishing their respective class, in other words, there is an underlying assumption that all binary classifiers are totally trustable and equally reliable, which does not always hold in multi-class cases as Yi Liu [17] shows through a simple example as in figure 4.

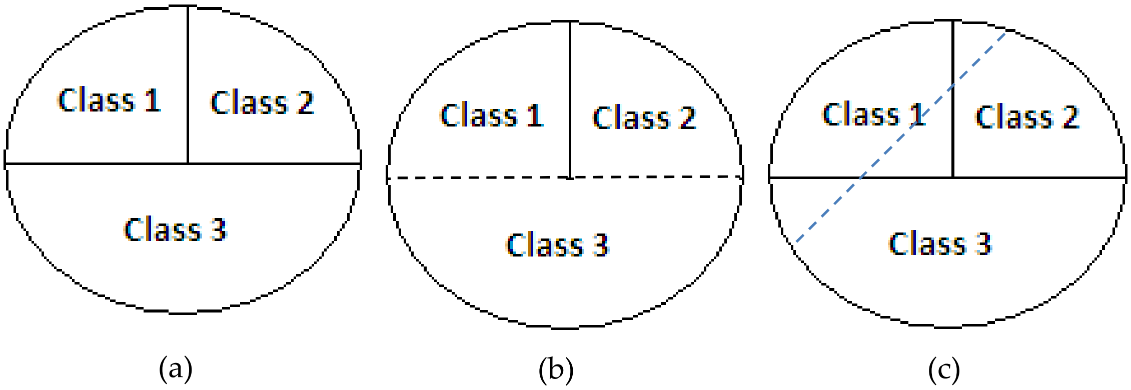


Figure 4. (a) Three classes problem and respective boundaries; (b) binary classifier that distinguishes well class 3 from all others (dashed line); (c) binary classifier that does not distinguish well class 1 from all others (dashed line). The example was taken from [15].

The same error occurs with the binary classifier for class 2 and so, the multi-class classifier based on these three binary classifiers would not provide good accuracy. In order to mitigate such problem, Liu [15] suggests two reliability measures: SRM – static reliability measure and DRM – dynamic reliability measure.

3.1.1. Static reliability measure

As pointed out by Vapnik [17] and Cortes [1], a small training set error does not guarantee a small generalization error when the number of training samples is relative small with respect to the feature vector **x** dimension. SVM training is done minimizing the objective function and as the objective function becomes smaller, smaller also becomes the generalization error. Based on this fact, Liu rewrites the objective function as seen in (28) and proposes the SRM as in (29).

$$Obj = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(1 - y_i D(x_i))_+ \tag{28}$$

Where $(u)_+ = u$ if $u > 0$ and 0 if $u \leq 0$.

$$\lambda_{SRM} = \exp\left(-\frac{1/2\|w\|^2 + C\sum_{i=1}^{N}(1 - y_i D(x_i))_+}{\sigma} \right) \tag{29}$$

Where $D(x_i) = w^T x_i + b$, and the parameter $\sigma = CN$ is a normalization factor to offset the effect of the different regularization parameter C and training size N. This λ_{SRM} metric is reduced to (30) for those linearly separable cases where $(1 - y_i D(x_i))_+ = 0$ for all training samples.

$$\lambda_{SRM} = \exp\left(-\frac{\|w\|^{\;2}}{2CN} \right) \tag{30}$$

From (28) we notice that $2/\|w\|^2$ is the classification margin. Small $\|w\|$ corresponds to large margin and more accurate classifier. Small $\|w\|$ also corresponds to larger reliability measure λ_{SRM}.

3.1.2. Dynamic reliability measure

The basic idea, differently of the static measure that is global over the whole training samples, is to estimate the classifier's reliability in a local region of feature space surrounding the test sample **x**. The 'k' surrounding samples of **x** are denoted by $N_k(x)$.

Suppose A(**x**) ∈ {1, -1} is the class label assigned to **x** by a SVM classifier and let $N_k^{A(x)}(x)$ denote the set of the training samples that belong to the set of 'k' nearest neighbors of **x** and are classified to the same class of **x**. Now, rewriting equation (28) as in (31)

$$Obj = \sum_{i=1}^{N}\frac{1}{2N}\|w\|^2 + C\sum_{i=1}^{N}(1 - y_i D(x_i))_+ = \sum_{i=1}^{N} Obj(x_i) \tag{31}$$

Liu formulate the local version of OBJ as in (32)

$$Obj_{local} = \sum Obj(\hat{x}_i) = \sum_{i=1}^{k_x}\left(\frac{1}{2N}\|w\|^2 + C\sum_{i=1}^{N}(1 - y_i D(x_i))_+ \right) \tag{32}$$

Where $\hat{x}_i \in N_k^{A(x)}(x), (\hat{x}_i, \hat{y}_i)$ is the training pair, and k_x is the number of training samples in the set $N_k^{A(x)}(x)$. And the dynamic reliability measure becomes as in (33).

$$\lambda_{DRM}(x) = \exp\left(-\frac{Obj_{local}}{C \cdot k_x} \right) \tag{33}$$

3.1.3. SRM and DRM decision rule

For a test sample **x**, assuming 'M' trained support vector machines each with its decision function, we evaluate D(x) for each classifier and after, generate the corresponding soft decision output $y_i \in [-1,+1]$ assuming that all classifiers are completely trustable (34)

$$y_i = sign\left(D_i(x)\right)(1-\exp\left(\left|D_i\left(x\right)\right|\right) \tag{34}$$

Now, assuming that λ_i denotes either the SRM or DRM reliability measure we have (35)

$$\begin{aligned} &\tilde{y}_i = y_i \cdot \lambda_i \\ &and \\ &i^* = \underset{i=1\ldots M}{\arg\max}\, \tilde{y}_i \end{aligned} \tag{35}$$

Mota in [18] sees the same problem from a different point of view. According to them in the One-Against-All method, SVM binary classifiers are obtained by solving different optimization problems and the outputs from these binary classifiers may have different distributions, even when they are trained with the same set of parameters and so, comparing these outputs using equation (27) may not work very well.

The output mapping, as suggested in [18], tries to mitigate such problem normalizing the outputs of the binary classifiers in such way to make them comparable by the equation (27). Four strategies are suggested: MND, BND, DNCD and MLP, based, respectively, on distance normalization (the first three) and base on a neural network model (the last one).

3.1.4. MND output mapping strategy

Analyzing the histogram of the raw outputs (original outputs) from a typical SVM binary classifier (figure 5) we observe a bimodal distribution consisting of two normal functions each with different mean and standard deviation. Different binary classifiers show different values of mean and standard deviation which makes unfair to compare their outputs. Then, before applying equation (10), the outputs from each binary classifier are normalized in a way that they all provide a normal distribution with mean at −1 or +1 and a standard deviation equal to 1.

Using a validation data set the samples are grouped into two groups A_1 (the current class) and A_2 (all the other classes) and the respective output distribution mean and standard deviation are computed and then the normalized output is obtained by equation (36).

$$u_i = \frac{d'_{(y_i,-1)} + d'_{(y_i,+1)}}{2} \tag{36}$$

Where

$$d'_{(y_i,k)} = \frac{y_i - \mu_k}{\sigma_k}, k = \{-1,+1\} \tag{37}$$

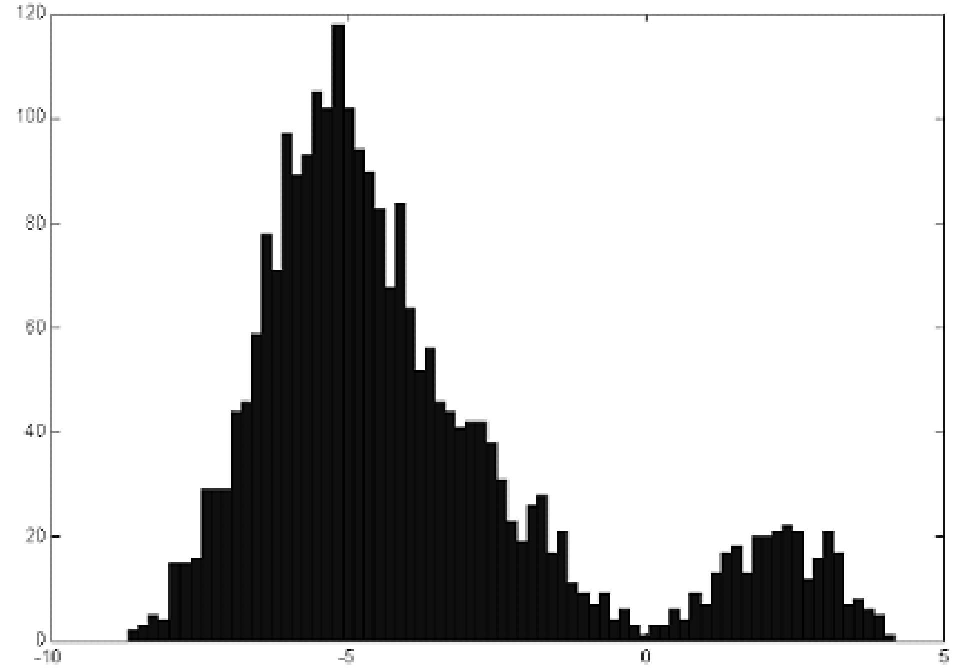


Figure 5. Output histogram of a binary one-against-all classifier

3.1.5. BND output mapping strategy

BND Strategy takes into account both normalized distances using the equation (38). When both distances $d'_{(y_i,k)}, k = \{-1,+1\}$ are positive (i.e., yi is on the right side of the centers of both normal functions) then ui is +1. When both distances are negative yi is on the left side of both centers and ui is −1, but when the distance signals are different, ui is between −1 and +1, closer to +1 if $d'_{(y_i,+1)}$ is greater than $d'_{(y_i,-1)}$, 0 when the distances are equal and closer to −1 otherwise.

$$u_i = \frac{d'_{(y_i,-1)} + d'_{(y_i,+1)}}{\left|d'_{(y_i,-1)}\right| + \left|d'_{(y_i,+1)}\right|} \tag{38}$$

3.1.6. DNCD output mapping strategy

Instead of using normalized distances (like in MND Strategy), DNCD Strategy builds a normalized output by joining the non-normalized distances and normalizing it by the distance between the centers of the normal functions as in equation (39).

$$u_i = \frac{d'_{(y_i,-1)} + d'_{(y_i,+1)}}{\left|\mu_{-1} - \mu_{+1}\right|} \tag{39}$$

3.1.7. MLP output mapping strategy

In this case, instead of having a function which maps each raw output y_i to a normalized output u_i, we have a function which maps the entire vector **y** into the vector **u**. The idea of this strategy is to implement the mapping function by using an MLP neural network, trained using a validation data set. The training samples are the outputs given by the multiclassifier for the validation data set. The expected outputs for those samples are the multiclassifier expected outputs, that is, a vector for which all positions have value −1, except for the one which corresponds to the class of that sample, whose value is +1.

Homogeneous M class multiclassifier is the one where its M binary classifiers are all trained with the same set of parameters. This approach, however, may not be the best option once the training of each classifier is independent and so, the chance is high to find a better set of classifiers if the search for different parameters is allowed in each case. But, in these cases, if a number ‘g’ of such parameters is used then the number of possible combinations of them is g^s and, obviously, even for reasonable values of ‘g’ the test for all possible combinations is impracticable.

One approach is to choose a subset of alternative parameters composition and train a set of L distinct homogeneous multiclass SVMs. The output mapping is then applied to each of the ‘L*s’ binary classifiers and the heterogeneous multiclassifier is formed by selecting the best binary classifier from the ‘L’ homogeneous multiclassifiers. The selection is done through the classification quality metric ‘q’ as in (40) computed from the confusion matrix of each binary classifier.

$$q_i = \frac{2M_{ii}}{\sum_{j=1}^{s} M_{ij} + \sum_{j=1}^{i} M_{ji}} \tag{40}$$

Where M_{ij} is the value of the i-th row and j-th column of the confusion matrix, which corresponds to the number of samples of class A_i that were missclassified as being of class A_j by the homogeneous multiclassifier. The more q_i approaches to 1 the better is the interaction of the i-th binary SVM among the other ones of the same homogeneous multiclassifier. Thus, not only we take into account the number of hits of an SVM, but also we penalize it for possible confusions in that multiclassifier. Finally, the heterogeneous multiclassifier is produced by the binary SVMs of greatest quality for each class.

3.2. The one-versus-one strategy

This method constructs one binary classifier for every pair of distinct classes and so, for M classes, a number of M*(M-1)/2 binary classifiers are constructed. The binary classifier A_{ij} is trained taking the examples from class ‘i’ as positive and the examples from class ‘j’ as negative. For a new example **x**, if classifier A_{ij} classifies it as class ‘i’, then the vote for class ‘i’ is added by one. Otherwise, the vote for class ‘j’ is increased by one. After each of the M*(M-

1)/2 binary classifiers makes its vote, the strategy assigns the current example to the class with the largest number of votes.

Two interesting variations for the One-Against-One strategy, not using maximum vote, were proposed, one by Hastie and Tibshirani [19] known as pairwise coupling and other by Platt [20] that is a sigmoid version of the same pairwise coupling approach suggested by Hastie. Another interesting variation of this pairwise is proposed by Moreira and Mayoraz [21].

3.2.1. Pairwise coupling

Considering that each binary classifier C_{ij} on a One-Against-One strategy provides a probabilistic output as $r_{ij} = prob(A_i / A_i \cup A_j), i \neq j$, Hastie and Tibshirani propose to combine them in order to obtain an estimation of the posterior probabilities for all classifiers together $p_i = prob(A_i / x), i = 1...M.$ To estimate the p_i's, M*(M-1)/2 auxiliary variables μ_{ij}'s related to the p_i's are introduced such as: $\mu_{ij} = p_i/(p_i + p_j)$ and then, p_i's are determined so that μ_{ij}'s are close to r_{ij}'s. Kullback-Leibler distance [22, 23] between r_{ij} and u_{ij} was chosen as the measure of closeness (41).

$$l_p = \sum_{i<j} n_{ij} \left(r_{ij} \log \frac{r_{ij}}{\mu_{ij}} + \left(1 - r_{ij}\right) \log \frac{1 - r_{ij}}{1 - \mu_{ij}} \right) \tag{41}$$

where n_{ij} is the number of examples that belongs to the union of both classes (A_i U A_j) in the training set. The associated score equations are (42).

$$\sum_{i \neq j} n_{ij} \mu_{ij} = \sum_{i \neq j} n_{ij} r_{ij}, \, i = 1...M, \, subject \; to \sum_{k=1}^{M} p_k = 1 \tag{42}$$

The p_i's are computed using the following iterative procedure:

1. Start from an initial guess of p_i's
2. Compute the corresponding μ_{ij}'s: $\mu_{ij} = \dfrac{p_i}{p_i + p_j}$
3. Repeat (i = 1 … M and so on) until the convergence is reached:

a. $p_i = p_i \dfrac{\sum_{i \neq j} n_{ij} r_{ij}}{\sum_{i \neq j} n_{ij} \mu_{ij}}$

b. Renormalize p_i's: $p_i = \dfrac{p_i}{\sum_{i=1}^{M} p_i}$

c. Recomputed μ_{ij}'s

3.2.2. Sigmoid pairwise coupling

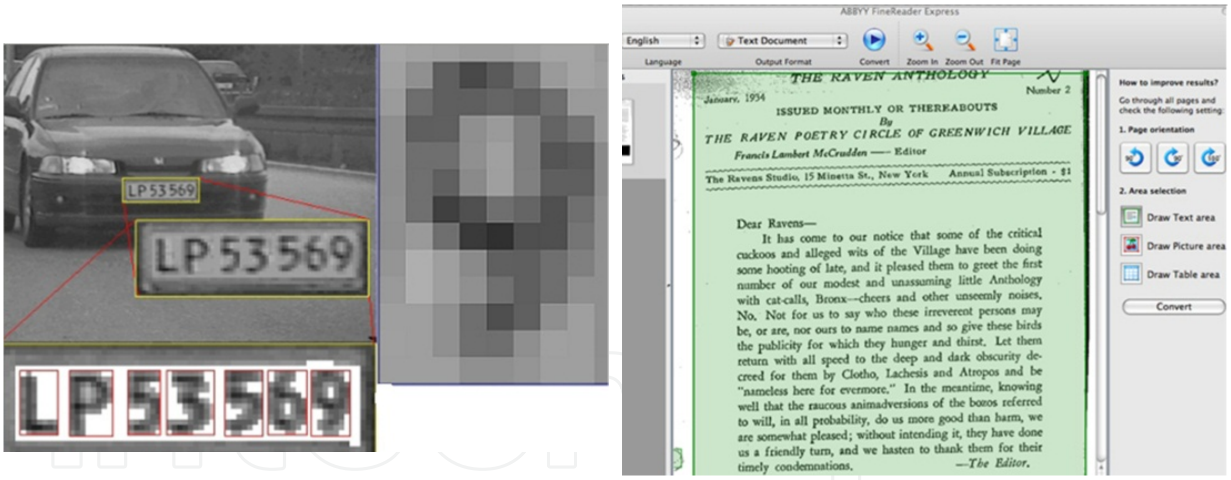
Platt criticized Hastie and Tibshirani's method of generating posterior class probabilities for each binary SVM, and suggested the use of a properly designed sigmoid applied to the SVM output to form these probabilities such as in (43).

$$\Pr(w_1 \mid x) = \frac{1}{1+e^{Af+B}} \tag{43}$$

Where 'f' is the output of the SVM associated with the example **x** and the parameters 'A' and 'B' are determined by the minimization of the negative log-likelihood function over the validation data. In [20] Platt suggests a pseudo-code for the determination of the parameters 'A' and 'B'.

4. Character recognition experiments

The ability to identify machine printed characters in an automated manner has obvious applications in numerous fields (figure 6). *Optical character recognition (OCR), as this field is commonly known,* has been a topic of interest for a long time since the late 1940's, when Jacob Rabinow started his work. Jacob was an engineer and inventor, he lived from 1910 to 1999 and during his life he earned 230 U.S. patents on a variety of mechanical, optical and electrical devices.



THE RAVEN ANTHOLOGY

January, 1934 Number 2

ISSUED MONTHLY OR THEREABOUTS
By
THE RAVEN POETRY CIRCLE OF GREENWICH VILLAGE
Francis Lambert McCrudden — Editor

The Ravens Studio, 15 Minetta St., New York Annual Subscription - $1

Dear Ravens—

It has come to our notice that some of the critical cuckoos and alleged wits of the Village have been doing some hooting of late, and it pleased them to greet the first number of our modest and unassuming little Anthology with cat-calls, Bronx—cheers and other unseemly noises. No. Not for us to say who these irreverent persons may be, or are, nor ours to name names and so give these birds the publicity for which they hunger and thirst. Let them return with all speed to the deep and dark obscurity decreed for them by Clotho, Lachesis and Atropos and be "nameless here for evermore." In the meantime, knowing well that the raucous animadversions of the bozos referred to will, in all probability, do us more good than harm, we are somewhat pleased; without intending it, they have done us a friendly turn, and we hasten to thank them for their timely condemnations. —*The Editor.*

(a) (b)

Figure 6. a) Example of a LPR – License Plate Recognition application; b) Example of a text reading from scanned paper

The earliest OCR machines were primitive mechanical devices with fairly high failure rates. As the amount of new written material increased, so did the need to process it all in a fast and reliable manner, and these machines were clearly not up to the task. They quickly gave way to computer-based OCR devices that could outperform them both in terms of speed and reliability.

Today there are many OCR devices in use based on a variety of algorithms. Despite the fact that these OCR devices can offer good accuracy and high speed, they are still far away compared to the performance reached by the human being. Many challenges are still opened not only with respect to the variety of scenarios, as well as, types of printed characters and handwritings, but also with respect to the accuracy by itself. There is no device able to recognize 100%, they always make mistake and, sometimes, bad mistakes like find a character that does not exist or recognize a complete different character than it really is (example: recognize as an 'M' what in fact is an 'S').

4.1. Remarks

The research field on automatic algorithms for character recognition is very large including different forms of characters like Chinese, Arabic and others; different origin like printed and handwritten and different approaches to obtain the character image like on line and off line.

The experiments on character recognition reported in the literature vary in many factors such as the sample data, pre-processing techniques, feature representation, classifier structure and learning algorithm. Only a reduced number of these works have compared their proposed methods based on the same set of characters. Obviously that this fact makes tough to get a fair comparison among the reported results.

Some databases were created and divulgated to the researcher's community with the objective to offer a generic and common set of characters to be used as patterns for the researches. Some of the most popular databases are CENPARMI, NIST, MNIST and DEVNAGARI.

License Plate and handwritten numeral recognition are on the most addressed research topics in nowadays and the experiments on handwritten numeral have been done basically using CENPARMI and NIST Special Database 19.

CENPARMI database, for example, contains 4,000 training samples and 2,000 test samples segmented from USPS envelope images. This set is considered difficult but it is easy to achieve in the literature recognition rates reported over 98%. Suen et al. reported accuracy of 98.85% by training neural networks on 450,000 samples [24] training it with 4,000 samples. Liu et al. report rates over 99% using polynomial classifier (PC) and SVMs [25], [26]. They report an accuracy of 99.58% using RBF SVM and 99.45% using Polynomial SVM. In [27] Ahmad et al. report the usage of a hybrid RBF kernel SVM and a HMM – Hidden Markov Model system over an online handwriting problem taken from the IRONOFF-UNIPEN database. The same authors in [28] report a work done on the recognition of words. Pal et al. also report in [29] the usage of a hybrid system based on SVM and MQDF – Modified Quadratic Discriminant Function for the problem of Devnagari Character Recognition. Arora et al., all from India, report in [30] a performance comparison between SVM and ANN – Artificial Neural Network on the problem of Devnagari Character Recognition.

License Plate recognition as well as off line handwritten recognition represents a very tough challenge for the researchers. There are a number of possible difficulties that the recognition algorithm must be able to cope with, which includes, for example: a) poor image resolution, usually because the camera is too far away from the plate; b) poor lighting and low contrast due to overexposure, reflection or shadows; c) object obscuring (part of) the plate, quite often a tow bar, or dirt on the plate; d) bad conservation state of the plate; e) Blurry images, particularly motion blur; and f) lack of global pattern, sometimes even inside a same country or state (figure 7).

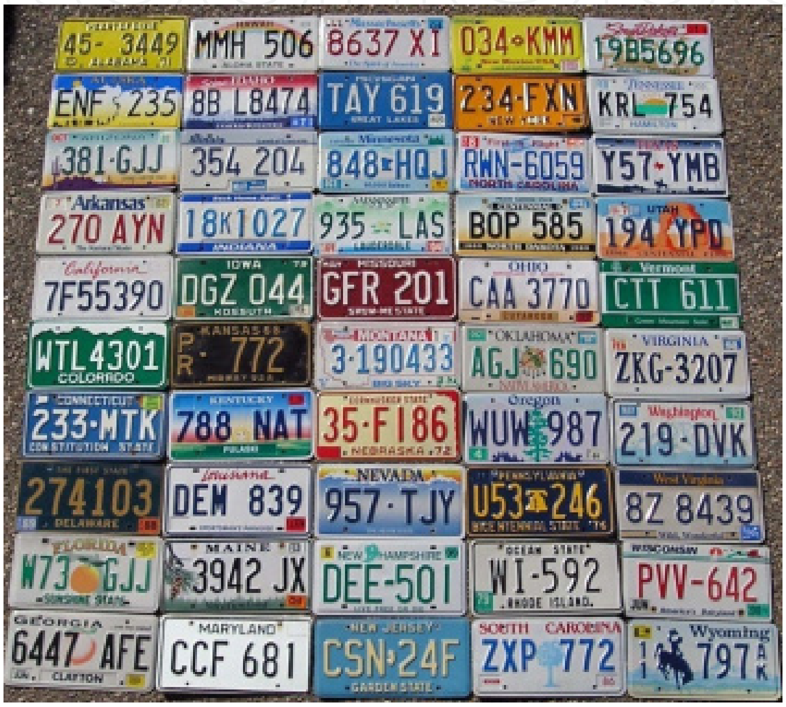
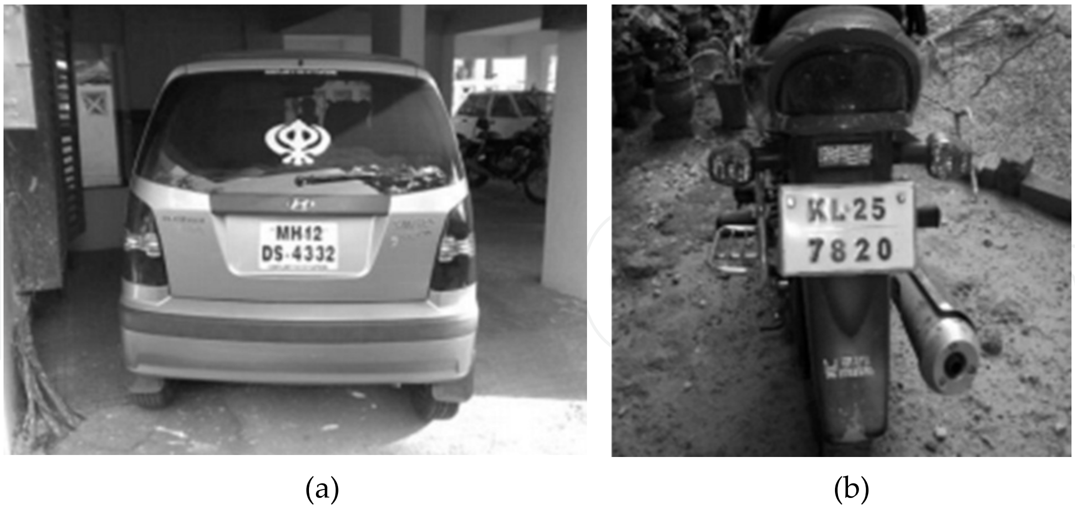


Figure 7. Example of License plate samples from 50 states of USA [31]

There is plenty of research work on this subject reported in the literature but the accuracy comparison among them is even more complex and difficult than the work done on handwritten. The accuracy not only depends on the type of the plates itself but also on the conditions on which the images were taken and on the level of the problems cited on previous paragraph. Waghmare et al. [32] report the use of 36 One-Against-All multiclass SVM classifier trained to recognize the 10 numeral and 26 letters from Indian plates (figure 8a). Parasuraman and Subin in [33] also report the usage of a Multiclass SVM classifier to recognize plates from Indian motorcycles (figure 8b). Other works on LPR can be found in [34 – 37].

In summary, character recognition is intrinsically a non linear and high dimensional problem. Among to the variety of OCR algorithms found in the literature, the SVM classifier is one of the most popular based on its good accuracy, high response speed and robustness. In the following subsections we describe some experiments in character recognition using both One-Against-All and One-Against-One multiclass SVMs.



(a) (b)

Figure 8. Indian plates for car (a) and motorcycle (b)

4.2. Using one-against-all strategy

The strategies proposed in [18] are here evaluated when applied to a problem of classifying characters extracted from vehicle plates. Two multiclass SVMs are trained: one to recognize the 10 digits (10 classes) and other to distinguish the 26 letters (26 classes).

For each group of characters, three data sets were formed based on the feature extraction used. In data set 1 (DS1) the feature vector has dimensionality of 288 formed by the 16 × 16 character bit matrix and 32 additional values from the character horizontal and vertical projections. Principal Component Analysis [38] reduced the original dimension to 124 (for digits) and 103 (for letters). Data sets 2 (DS2) and 3 (DS3) were generated respectively by 56 and 42 statistical moments extracted from the 16 x 16 character bit matrix.

Each data set was divided in three subsets: one for training, one for validation, and one for test. Table 1 shows how the samples were divided in these three subsets.

Subset	Digits	Letters
Training	2794	2088
Validation	2500	1875
Test	7500	5625
Total	12,794	9,588

Table 1. Number of samples for each data subset

4.2.1. Digit recognition

Based on two kernel functions and a set of different values for two variables as shown in table 2 (standard deviation for the Gaussian kernel and exponent order for the polynomial kernel), a set of 55 Homogeneous multiclassifiers were trained.

Kernel Function	Kernel Parameter	C value
Polynomial	[1, 4]	[0.1, 10]
Gaussian	[1.0, 3.0]	[5, 80]

Table 2. Number of samples for each data subset

The heterogeneous multiclassifier was formed with 10 binary classifiers selected each one of the 55 homogeneous multiclassifiers using the output mapping and confusion matrices as explained in previous section. The best results achieved for the test subsets of each data set are seen in Table 3. WTA-SVM is the common Winner-Takes-All strategy for One-Against-All approach.

Strategy	DS1	DS2	DS3
WTA-SVM	4.16%	2.95%	2.77%
MND	4.07%	**2.84%**	2.73%
BND	4.23%	2.91%	2.83%
DNCD	4.33%	3.05%	2.87%
MLP	**3.48%**	2.85%	**2.57%**

Table 3. Error results on the test sets

4.2.2. Letter recognition

The multiclassifier construction was also based on the same two kernel functions and the same two variables as shown in table 4 (standard deviation for the Gaussian kernel and exponent order for the polynomial kernel), a set of 30 Homogeneous multiclassifiers were trained.

Kernel Function	Kernel Parameter	C value
Polynomial	[2, 4]	[0.05, 5]
Gaussian	[1.0, 3.5]	[1, 20]

Table 4. Number of samples for each data subset

The heterogeneous multiclassifier was formed with 26 binary classifiers selected each one of the 30 homogeneous multiclassifiers using the output mapping and confusion matrices as explained in previous section. The best results achieved for the test subsets of each data set are seen in Table 5. WTA-SVM is the common Winner-Takes-All strategy for One-Against-All approach.

Strategy	DS1	DS2	DS3
WTA-SVM	5.24%	4.36%	3.29%
MND	5.74%	4.37%	3.57%
BND	6.24%	4.52%	3.72%
DNCD	5.64%	4.46%	3.45%
MLP	**4.46%**	**4.12%**	**2.88%**

Table 5. Error results on the test sets

4.3. Using one-against-one strategy

Experiments using One-Against-One RBF Kernel SVM are described in [39] for Brazilian plates, a total of 22464 numerals and 16848 letters were used for training and testing the system (Table 6). Brazilian plates show two different patterns (figure 9 – black characters over gray background and white characters over red background).

Subset	Digits	Letters
Training	3942	3024
Test	18522	13824
Total	22,464	16,848

Table 6. Number of samples for each data subset

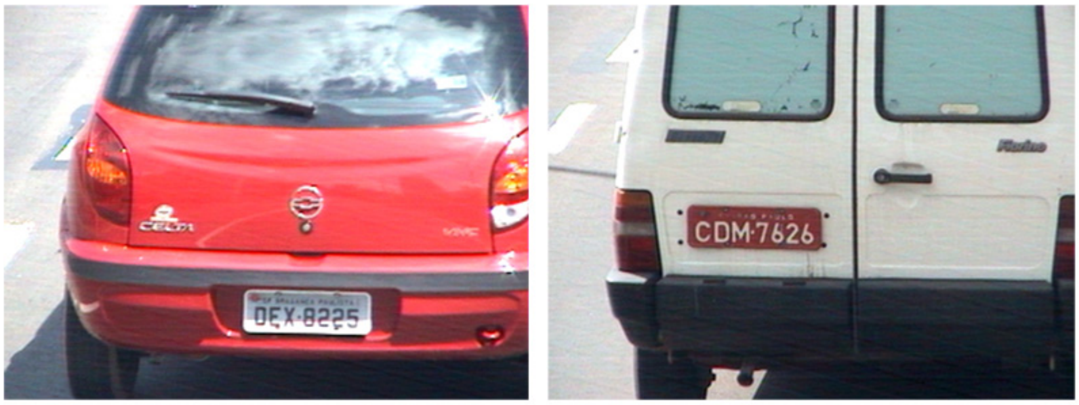


Figure 9. Brazilian patterns of plate

4.3.1. Digit recognition

As reported, an average accuracy of 99.61% was achieved (Table 7) and the worst performance occurred on the recognition of the number 8, which was misclassified 17 times from a total of 1725 samples (Table 8 – 7 times misclassified as 6 and 7 as 9).

	Correct Classification		**Error**	
Label	**Number**	**%**	**Number**	**%**
0	2222	99.64%	8	0.36%
1	2095	99.90%	2	0.10%
2	1840	99.67%	6	0.33%
3	1799	99.89%	2	0.11%
4	1716	99.42%	10	0.58%
5	1700	99.77%	4	0.23%
6	1751	99.38%	11	0.62%
7	1825	99.67%	6	0.33%
8	1708	99.01%	17	0.99%
9	1793	99.61%	7	0.39%
Total	18449		73	
Average		**99.61%**		**0.39%**

Table 7. Percentage of classification for Digits

SVM \ Target	0	1	2	3	4	5	6	7	8	9
0	2222	0	0	0	2	0	0	0	0	0
1	0	2095	0	0	0	0	0	0	0	0
2	1	0	1840	0	1	2	0	0	0	2
3	0	0	0	1799	0	1	0	0	1	0
4	1	0	0	0	1716	1	6	1	0	1
5	0	1	0	0	0	1700	2	0	0	1
6	1	0	0	0	1	2	1751	0	4	3
7	0	0	2	1	0	0	0	1825	0	3
8	2	0	0	0	0	1	7	0	1708	7
9	1	0	0	2	0	3	0	0	1	1793

Table 8. Digits confusion matrix

4.3.2. Letter recognition

As reported, an average accuracy of 98.60% was achieved (Table 9) and the worst performances showed by letters D (89.76%), O (93.31%) and Q (94.51%).

	Correct Classification		Error			Correct Classification		Error	
Label	Number	%	Number	%	Label	Number	%	Number	%
A	408	100.00%	0	0.00%	N	719	98.76%	9	1.24%
B	450	98.90%	5	1.10%	O	669	93.31%	48	6.69%
C	576	99.31%	4	0.69%	P	440	99.32%	3	0.68%
D	298	89.76%	34	10.24%	Q	379	94.51%	22	5.49%
E	221	98.66%	3	1.34%	R	401	99.01%	4	0.99%
F	177	98.88%	2	1.12%	S	327	99.09%	3	0.91%
G	250	98.43%	4	1.57%	T	340	99.71%	1	0.29%
H	309	98.10%	6	1.90%	U	531	98.88%	6	1.12%
I	168	97.67%	4	2.33%	V	374	99.73%	1	0.27%
J	337	99.70%	1	0.30%	W	284	98.95%	3	1.05%
K	1590	99.69%	5	0.31%	X	287	98.97%	3	1.03%
L	2925	99.59%	12	0.41%	Y	283	99.30%	2	0.70%
M	349	97.76%	8	2.24%	Z	538	99.81%	1	0.19%
Total	13630		194						
Average		98.60%		1.40%					

Table 9. Percentage of classification for Digits

Table 10 shows and explains the reasons for such reduced performance in comparison to the other 23 letters. The fact is that these three letters show very similar visual aspect and the SVM misclassified 23 letters ‘O’ as ‘D’, 26 ‘D’ as ‘O’, 17 ‘Q’ as ‘O’ and 18 ‘O’ as ‘Q’.

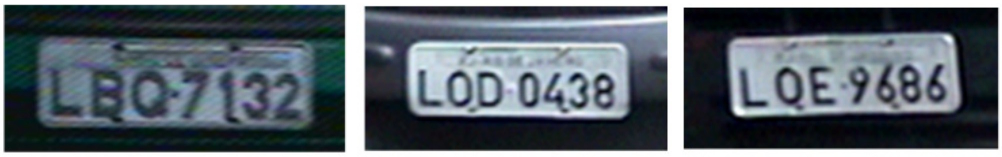


Figure 10. Similarity among letters D, O and Q

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	Z
B			2									1					1	1					
C					1	1					1					1							
D	3	3												**23**		3							
E					2													1					
F				2																			
G		3															1						
H												1	1		1		1					2	
I											3											1	
J								1															
K	1																		1		1	2	
L		1					2	6							1					2			
M	1							1		3			2									1	
N							1			1		3									1	3	
O	1	3	**26**												1	**17**							
P	1				2																		
Q		1	1			2								**18**									
R															2							2	
S	1									1							1						
T																							1
U							1				1			2		2							
V										1													
W						2		1															
X						3																	
Y											1								1				
Z									1														

Table 10. Letters confusion matrix

Author details

Antonio Carlos Gay Thomé
Federal University of Rio de Janeiro, Brasil

5. References

[1] C. Cortes and V. N. Vapnik, Support vector networks. Machine Learning, vol. 20, no. 3, pp. 273-297, 1995.

[2] Fisher, R. A.. *The use of multiple measurements in taxonomic problems*. Annals of Eugenics, 7, 111–132, 1936.

[3] Rosenblatt, Frank. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington DC: Spartan Books, 1962.

[4] Vapnik, V., and A. Lerner. *Pattern recognition using generalized portrait method*. Automation and Remote Control, 24, 774–780, 1963.

[5] Aizerman, M. A., E. M. Braverman, and L. I. Rozonoer. *Theoretical foundations of the potential function method in pattern recognition learning*. Automation and Remote Control, 25, 821–837, 1964.

[6] Cover, Thomas M.. *Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition*. IEEE Transactions on Electronic Computers, 14, 326–334, 1965.

[7] Vapnik, V. N., and A. Ya. Chervonenkis. *Teoriya raspoznavaniya obrazov: Statisticheskie problemy obucheniya*. (in Russian) [Theory of pattern recognition: Statistical problems of learning]. Moscow: Nauka, 1974.

[8] Vapnik, V.. *Estimation of Dependences Based on Empirical Data* [in Russian]. Moscow: Nauka, 1979.

[9] Boser, Bernhard E., Isabelle M. GUYON, and Vladimir N. VAPNIK. *A training algorithm for optimal margin classifiers*. In: COLT ’92: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. New York, NY, USA: ACM Press, pp. 144–152, 1992.

[10] Theodoridis, S. and Koutroumbas, K., Pattern Recognition, 4th edition, Elsevier, 2009.

[11] Bishop, C. M., Pattern Recognition and Machine Learning, Springer, ISBN-13: 978-0-387-31073-2, 2006.

[12] C.-W. Hsu and C.-J. Lin, “A comparison of methods for multiclass support vector machines,” IEEE Transactions on Neural Networks, vol. 13, no. 2, pp. 415–425, 2002.

[13] J. C. Platt, N. Cristianini, and J. Shawe-taylor, “Large margin dags for multiclass classification,” in Advances in Neural Information Processing Systems. MIT Press, pp. 547–553, 2000.

[14] E. L. Allwein, R. E. Schapire, and Y. Singer, “Reducing multiclass to binary: A unifying approach for margin classifiers,” The Journal of Machine Learning Research, vol. 1, pp. 113–141, 2001.

[15] Yi Liu, One-against-all multi-class SVM Classification using reliability measures, Neural Networks, IJCNN '05, 2005.

[16] R. M. Rifkin and A. B. R. Klautau, “In defense of one-vs-all classification,” The Journal of Machine Learning Research, vol. 5, pp. 101–141, 2004.

[17] V. N. Vapnik, An Overview of Statistical Learning Theory, IEEE Transactions on Neural Networks, vol. 10, no. 5, pp.. 988-999, 1999.

[18] Thiago C. Mota and Antonio C. G. Thomé, One-Against-All-Based Multiclass SVM Strategies Applied to Vehicle Plate Character Recognition, IJCNN, 2009.

[19] Hastie, T., Tibshirani, R., Classification by pairwise coupling. The Annals of Statistics, vol 26, nr. 2, 451-471, 1998.

[20] Platt, J., Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. Advances in Large Margin Classifiers, 61-74, MIT Press, 1999.

[21] Moreira, M., Mayoraz, E., Improved Pairwise Coupling Classification with Correcting Classifiers, Tenth European Conference on Machine Learning, Chemnist – Germany, 1998.

[22] Kullback, S.; Leibler. "On Information and Sufficiency". *Annals of Mathematical Statistics* 22 (1): 79–86, 1951.

[23] Kullback, S.; Burnham, K. P.; Laubscher, N. F.; Dallal, G. E.; Wilkinson, L.; Morrison, D. F.; Loyer, M. W.; Eisenberg, B. et al. "Letter to the Editor: The Kullback–Leibler distance". *The American Statistician* 41 (4): 340–341, 1987.

[24] Suen, C.Y., K. Kiu, N.W. Strathy, Sorting and recognizing cheques and financial documents, Document Analysis Systems: Theory and Practice, S.-W. Lee and Y. Nakano (eds.), LNCS 1655, Springer, pp. 173-187, 1999.

[25] C.-L. Liu, K. Nakashima, H. Sako, H. Fujisawa, Handwritten digit recognition: benchmarking of state-of-the-art techniques, Pattern Recognition, 36(10): 2271-2285, 2003.

[26] C.-L. Liu, K. Nakashima, H. Sako, H. Fujisawa, Handwritten digit recognition: investigation of normalization and feature extraction techniques, Pattern Recognition, 37(2): 265-279, 2004.

[27] Ahmad, A. R., Viard-Gaudin, C., Khalid, M. and Yusof, R., Online Handwriting Recognition using Support Vector Machine, Proceedings of the Second International Conference on Artificial Intelligence in Engineering & Technology, Kota Kinabalu, Sabah, Malaysia, August 3-5 2004.

[28] Ahmad, A. R., Viard-Gaudin, C., Khalid, M., Lexicon-based Word Recognition Using Support Vector Machine and Hidden Markov Model, 10th International Conference on Document Analysis and Recognition, 2009.

[29] Pal, U., Chanda, S., Wakabayashi, T. and Kimura, F., Accuracy Improvement of Devnagari Character Recognition Combining SVM and MQDF.

[30] Arora, S., Bhattacharjee, D., Nasipuri, M., Malik, L., Kundu, M. and Basu, D. K., Performance Comparison of SVM and ANN for Handwritten Devnagari Character Recognition, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 6, May 2010.

[31] Li, X., Vehicle License Plate Detection and Recognition, Master Science Thesis presented to the Faculty of the Graduate School at the University of Missouri, 2010.

[32] Waghmare, S. K. and Gulve, V. N. N., Automatic Number Plate Recognition (ANPR) System for Indian Conditions using Support Vector Machine (SVM), International Journal of Computer Science and its Applications, 2010.

[33] Parasuraman, Kumar and Subin, P. S. SVM Based License Plate Recognition System, IEEE International Conference on Computational Intelligence and Computing Research, 2010.

[34] Abdullah, S. N. H. S., PirahanSiah, F., Abidin, N. H. H. Z., and Sahran, S., Multi-threshold approach for license plate recognition system, World Academy of Science, Engineering and Technology 72, 2010.

[35] Tsai, I., Wu, J., Hsieh, J. and Chen, Y., Recognition of Vehicle License Plates from a Video Sequence, IAENG International Journal of Computer Science, 36:1, IJCS_36_1_04, 2004.

[36] Abdullah, S. N. H. S., Intelligent License Plate Recognition System Based on Multi Feature Extractor and Support Vector Machine, Master Science Thesis at Faculty of Electrical Engineering – University of Technology of Malaysia, 2009.

[37] K P Tee, Etienne Burdet, C M Chew, Theodore E Milner, One-Against-All-Based Multiclass SVM Strategies Applied to Vehicle Plate Character Recognition, 2009 International Joint Conference on Neural Networks (2009), Volume: 90, Issue: 4, Publisher: Ieee, Pages: 2153-2159, ISBN: 9781424435531.

[38] I. T. Jolliffe, Principal Component Analysis. New York, NY, USA: Springer-Verlag, 1986.

[39] Medeiros, S., SVM Applied to License Plate Recognition, Technical Report, Federal University of Rio de Janeiro, Computer Science Department, Brazil, 2011.