Published in Image Processing On Line on YYYY–MM–DD.
Submitted on YYYY–MM–DD, accepted on YYYY–MM–DD.
ISSN 2105–1232 © YYYY IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://www.ipol.im/

A MATLAB SMO Implementation to Train a SVM Classifier: Application to Multi-Style License Plate Numbers Recognition

Pablo Negri[1,2]

[1] CONICET, Godoy Cruz 2290, Buenos Aires, Argentina
[2] Instituto de Tecnologia, Universidad Argentina de la Empresa (UADE), Lima 717, Buenos Aires, Argentina
(pnegri@uade.edu.ar)

PREPRINT January 5, 2016

Abstract

This paper implements the Sequential Minimimal Optimization (SMO) algorithm proposed by Platt, for Support Vector Machine (SVM) training. The code is written in MATLAB following the pseudo-code of the Platt's work. The classification is applied on a multi-style license plate recognition system. The features extracted from the characters (numbers) are based on Histograms of Oriented Gradient. This allows to be robust against changes on character/background colors.

Source Code

The source code implements the Platt's SMO algorithm to train and test a SVM classifier on a multi-style license plate character dataset. Compilation and usage instruction are included in the `README.txt` file of the archive. The online demo allows to test the character (numbers) recognition system, from license plate images.

Supplementary Material

The supplementary files of the work includes a dataset of license plate numbers from four countries having different fonts, and character/background colors.

Keywords: Sequential Minimal Optimization, Support Vector Machine, Multi-Style License Plate Recognition, Histogram of Oriented Gradient

1 Introduction

The Sequential Minimal Optimization (SMO) [8] can be considered as the simplest algorithm to train a Support Vector Machine (SVM) classifier. It employs the *divide and conquer* approach to solve analytically a large quadratic programming (QP) optimization problem. This reduction on the complexity has several advantages, saving time-processing and memory-consuming. aIt is also a very interesting implementation that can be employed for pedagogic purposes, because the variables of the iterative algorithm can be easily accessed and interpreted on the learning process.

This article a multi-class SVM classifier using SMO to address multi-style license plate numbers recognition problem. It is employed a One-Against-All approach to classify the incoming characters from "0" to "9". In order to be robust to changing colors and backgrounds, the features describing the characters shape consist on Histograms of Oriented Gradients (HoG) [1], as was proposed on Gómez et al. work [4].

2 Histograms of Oriented Gradient Feature Space

The Histograms of Oriented Gradient (HoG) [1] is a widely used feature space which is employed to sucesfully recognize different kind of classes, as pedestrians, vehicles, etc. It compute the gradient of the image, here it is used the 3x3 size Sobel filter. Then, the histograms are determined by accumulating the gradient magnitude of each pixel by their orientation value. The orientation of pixels are quantized to integer values between 0 and $D-1$ (here $D=6$) using modulo π instead of modulo 2π. In that way, the HoG feature set is independent of the character and background colors. Each HoG feature describing a region of the image is a histogram of D bins.

Each histogram j is defined as : $h_j(x_j, y_j, s_j, r_j)$, where r_j is the type of rectangle, s_j is the scale and (x_j, y_j) is its position in the window. The types of rectangles depend on the $(width, height)$ ratio which can be (s, s), $(s, 2 \cdot s)$, $(2 \cdot s, 1)$. We have a total of four scales : $s : \{4, 6, 8\}$. Each histogram h_j is computed as follows:

- all the pixels within the rectangle r are traversed,
- the histogram h_j accumulates the value on the gradient magnitude at this pixel on the bin corresponding to the quantized orientation (the number of quantized histogram bins is D),
- once all the pixels are evaluated, the *bin* values are normalised to obtain their sum equals to 1.

Fig. 1 presents two samples from an Argentinean and a USA license plate HoG features computed on two different rectangles. Green rectangle correspond to a square feature and red rectangle is a vertical rectangle feature. As can be seen, both HoG features are very similar for the two different license plate numbers. Green HoG features have the fourth bin (horizontal direction) as the highest value. Also, the red HoG features have vertical direction, the first bin, with the greatest value.

To accelerate the computation of the histograms it is employed an intermediate representation of the input image called Integral Histogram [9], which uses the Integral Image [14]. The Integral Histogram is a three dimensional table (the third dimension corresponds to orientations) allowing to accumulate gradient magnitude for a each orientation in a rectangular region with only four references to the Integral Histogram. In this way, each HoG feature can be built with *4xD* references in the Integral Histogram.

The complete HoG feature space of one character is a vector $\mathbf{x}$ of 871 concatenated histograms: $\mathbf{x} = \{h_1, ..., h_8 71\}$. This vector, consisting of $871xD$ values $\in \Re$ is the input of the SVM multi-class classifiers.

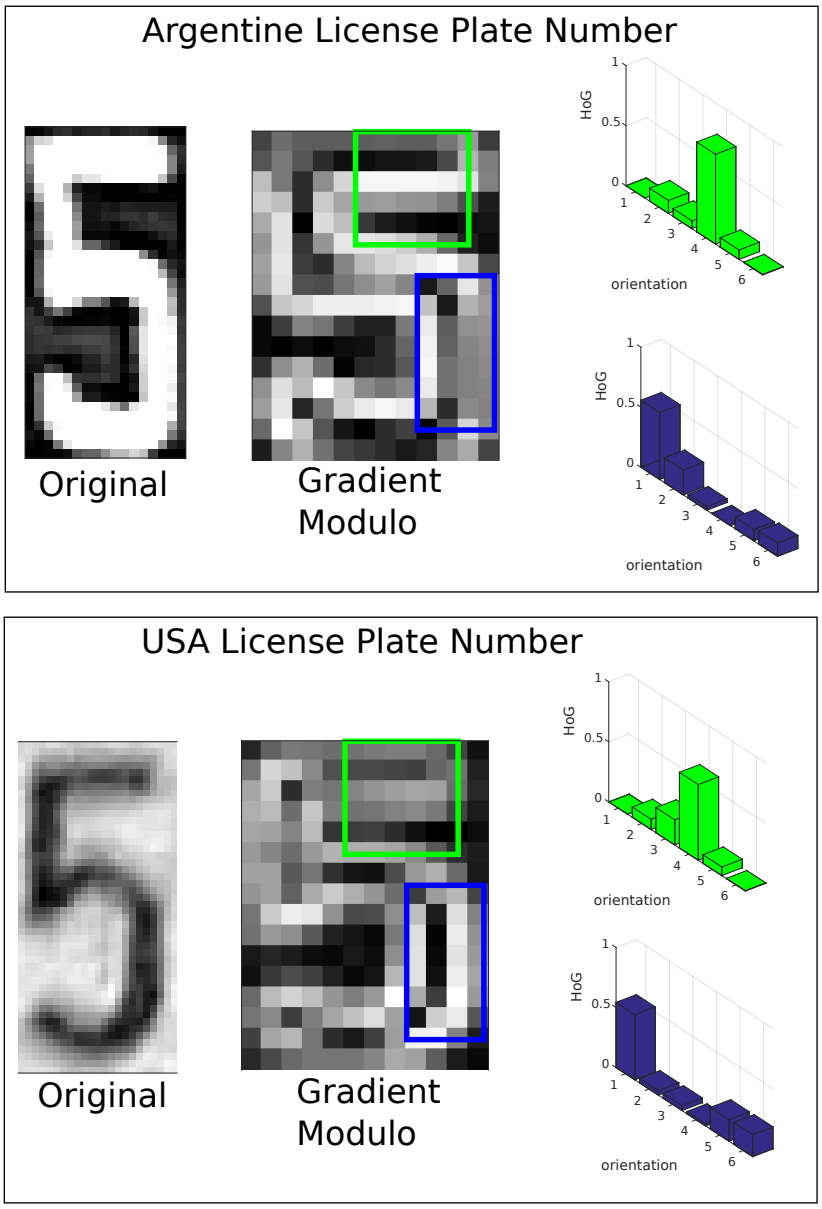

Figure 1: The figure examples HoG features for two samples from the two different datasets.

3 SVM-SMO Implementation

This section starts with a brief introduction to the SVM classification in order to define several entities that will be required later. The second part, details the SMO implementation code.

3.1 SVM Introduction

In 1995 Vapnik [13] introduces Support Vector Machines. It is an optimization algorithm seeking to find the hyperplane with maximum margin discriminating two classes on a dataset, as shown Fig. 2. For separable datasets, a linear hyperplane is always found. The margin is defined as the minimal distance between the nearest positive and negative samples and this hyperplane.

Let $\{\mathbf{x}_i, y_i\}_{i=1,\ldots,N}$ be a training dataset of size N, with $\mathbf{x}_i \in \Re^d$ the input vector of sample i having size d, and y_i their label that takes two possible values: -1 and 1. The formula of the linear

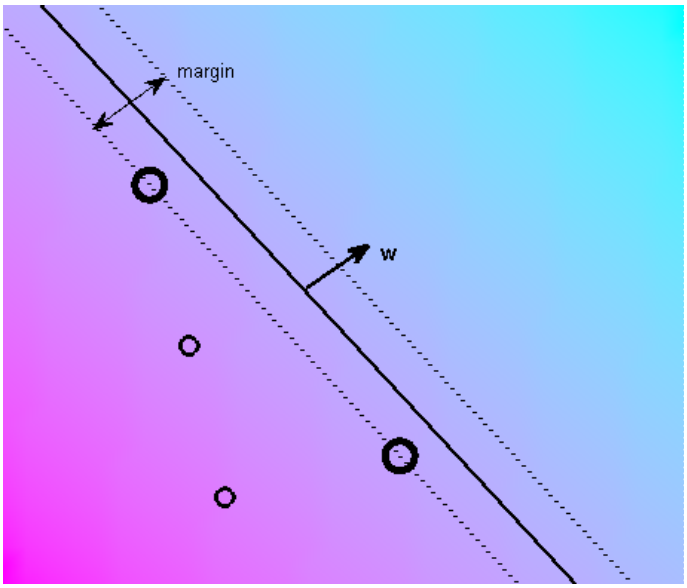

Figure 2: The figure shows a toy sample of the SVM margin and hyperplane definition on a two class problem.

hyperplane is:

$$f = \mathbf{w} \cdot \mathbf{x}_i - b \tag{1}$$

where $\mathbf{w}$ is the normal vector to the hyperplane, and b is a constant factor. The margin m is defined as:

$$m = \frac{1}{||\mathbf{w}||} \tag{2}$$

The separating hyperplane is the plane f=0, and the nearest samples lie on the plane $f = \pm 1$. The optimization problem it is then stated as a margin maximization:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 \text{ subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \forall i \tag{3}$$

which can be converted into a dual form where an objective function Ω only depend on a set of Lagrange multipliers α_i:

$$\min_{\alpha} \Omega(\alpha) = \min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \alpha_i \alpha_j - \sum_{i=1}^{N} \alpha_i \tag{4}$$

The associated constraints for the multipliers are:

- $\alpha_i \geq 0, \forall i$,
- $\sum_{i=1}^{N} y_i \alpha_i = 0$

There is a one-to-one relationship between each α_i and the training samples $\mathbf{x}_i$. The parameters $\mathbf{w}$ and b of the hyperplane can be computed using the Lagrange multipliers as:

$$\mathbf{w} = \sum_{i=1}^{N} y_i \alpha_i \mathbf{x}_i, \quad b = \mathbf{w} \cdot \mathbf{x}_k - y_k \quad \text{for some } \alpha_k > 0 \text{ (the support vectors)} \tag{5}$$

Most of the classification problems are not necessarily separable in the feature space. It is then mandatory to introduce slacks variables ζ_i allowing, but penalizing, the failure of the hyperplane by some samples placed on the wrong side. The modification on the optimization problem is:

$$\min_{\mathbf{w},b,\zeta} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\zeta_i \text{ subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i - b) \geq 1 - \zeta_i, \forall i \tag{6}$$

where C is a parameter which trade off wide margin with a small number of wrongly placed samples. The inequality constrains of the multipliers becomes:

$$0 \geq \alpha_i \geq C, \forall i \tag{7}$$

It is also introduced a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ which measures the similarity between the couple $(\mathbf{x}_i, \mathbf{x}_j)$. Examples of kernel functions include non-linear functions as Gaussians, polynomials, and neural networks, or a simple linear dot product. Their use generalize the optimization problem and the SVM classifiers output is computed by:

$$f(\mathbf{x}_t) = \sum_{j=1}^{N} y_j\alpha_j K(\mathbf{x}_t, \mathbf{x}_j) - b \tag{8}$$

where $\mathbf{x}_t$ is a test sample, and $\mathbf{x}_j$ are the stored training samples.

The dual objective function Ψ using the kernel function is still quadratic in α:

$$\min_{\alpha} \Psi(\alpha) = \min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)\alpha_i\alpha_j - \sum_{i=1}^{N}\alpha_i, \tag{9}$$

subject to

$$0 \leq \alpha_i \leq C, \forall i \tag{10}$$

$$\sum_{i=1}^{N} y_i\alpha_i = 0 \tag{11}$$

3.2 Sequential Minimal Optimization: Two Variables Analytic Solution

SMO is a simple algorithm that avoid numerical QP optimization solving a two Lagrange multipliers optimization problem at each iteration. At every step, SMO find the optimal value for these multipliers, and update the SVM framework.

The advantage of SMO is that for two Lagrange multipliers, the optimization sub-problem can be solved analytically without the use of QP.

For variables α_1 and α_2, the optimization of the objective function $\Psi(\alpha)$ from equation 9 becomes [11]:

$$\min_{\alpha_1,\alpha_2} \frac{1}{2}(\alpha_1^2 K_{11} + 2\alpha_1\alpha_2 K_{12} + \alpha_2^2 K_{22}) - \alpha_1 - \alpha_2 \tag{12}$$

subject to

$$0 \leq \alpha_1 \leq C, \text{ and } 0 \leq \alpha_2 \leq C \tag{13}$$

$$s\alpha_2 + \alpha_1 = \gamma \tag{14}$$

$$\alpha_1 = \gamma - s\alpha_2 \tag{15}$$

where $s = y_1 \cdot y_2$. Then, it can be removed α_1 and their constrains from the optimization problem and solved for α_2. Eq. 12 is rewritten using eq. 15:

$$\min_{\alpha_1} \frac{1}{2}\alpha_2^2(K_{11} + K_{22} - 2sK_{12}) + \alpha_2(2\gamma K_{12} - 2s\gamma K_{22} + s - 1) + \gamma^2 K_{22} - \gamma \tag{16}$$

Constrains on α_2 are also related to constrains on α_1, from equations 13 and 15:

$$0 \leq \alpha_2 \leq C, \text{ and } \gamma - C \leq \alpha_2 \leq \gamma \tag{17}$$

and can be stated as $L \leq \alpha_2 \leq H$.

The unconstrained objective function can be written as:

$$\frac{\chi}{2}\alpha_2^2 - \zeta\alpha_2 = 0 \tag{18}$$

$$\text{with } \zeta = 2s\gamma K_{22} - 2\gamma K_{12} - s + 1 \tag{19}$$

$$\text{and } \chi = K_{11} + K_{22} - 2sK_{12} \tag{20}$$

where the constant term was ignored. The minimum of α_2 is placed at $\chi^{-1}\zeta$, and, in order to assure that the solution is optimal for the constrains $[L, H]$, the unconstrained solution has to be clipped into the interval. The value of α_2 is computed considering two cases.
Case 1: $\chi = 0$

$$\alpha_2 = \begin{cases} H & \text{if } \zeta > 0 \\ L & \text{otherwise} \end{cases} \tag{21}$$

Case 2: $\chi > 0$

$$\alpha_2 = min(max(L, \chi^{-1}\zeta), H) \tag{22}$$

The case $\chi < 0$ occurs when two training samples have the same feature vector. To avoid this situation, a preliminary step eliminating duplicated inputs is then necessary.

The boundaries of the interval, L and H are obtained from eq. 17:

$$L = \begin{cases} max(0, s(\gamma - C)) & if s > 0 \\ max(0, s\gamma) & otherwise \end{cases} \tag{23}$$

$$H = \begin{cases} min(C, s\gamma) & if s > 0 \\ min(C, s(\gamma - C)) & otherwise \end{cases} \tag{24}$$

The value of α_1 cannot be deduced directly from eq. 15. It is necessary to develop an iterative algorithm updating the values of the Lagrange multipliers computing the minimal values along the direction of the constrains:

$$\alpha_2^{new} = \alpha_2^{old} + \frac{y_2(E_1 - E_2)}{\chi} \tag{25}$$

where $E_i = f_i - y_i$ is the error evaluating the ith sample with eq. 8. This value of α_2^{new} has to be *clipped* to assure that the value will lie between the constrains:

$$\alpha_2^{new,clipped} = \begin{cases} H & if \quad \alpha_2^{new} \geq H \\ L & if \quad \alpha_2^{new} \leq L \\ \alpha_2^{new} & otherwise \end{cases} \tag{26}$$

Finally, the value of α_1^{new} is then obtained from $\alpha_2^{new,clipped}$ as:

$$\alpha_1^{new} = \alpha_1^{old} + s(\alpha_2^{old} - \alpha_2^{new,clipped}) \tag{27}$$

3.3 Sequential Minimal Optimization Algorithm Implementation

The article [8] proposes a pseudo-code to solve SMO algorithm. It consist of three routines following step-by-step the methodology. Here, the pseudo-code is developed, explained in detail, and associated with the MATLAB code.

The Main Routine 1 initializes the SVM training algorithm for a two classification problem. The inputs of the procedure are the following:

- Training Dataset: it is composed of N pairs $(\mathbf{x}_i, y_i)$, where N is the length of the dataset, $\mathbf{x}_i$ is the feature vector of sample i, and y_i is the target of sample i corresponding to the following labels: $1, -1$, indicating to which class it belongs.
- Kernel Matrix Function $\mathbf{K}$: this is a two dimensional NxN matrix. Each element (i, j) of the matrix consist of the output of the non-linear kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$. For the license plate number recognition, it was employed a polynomial kernel function: $K(\mathbf{x}_i, \mathbf{x}_j) = 1 + \mathbf{x}_i \cdot \mathbf{x}_j$.
- C parameter: the SVM training is very sensible to this parameter. Generally, their value is computed by k-fold cross validation approach, choosing the value which maximizes the results on a validation set. In this work, the value of C is fixed to 10.
- b parameter: bias threshold parameter of the SVM hyperplane. It is initialized to 0.
- $\mathbf{E}$ error vector: this vector stores the errors of the training samples: $E_i = f_i - y_i$, where f_i is computed using equation 8. The initial values of the elements are: $E_i = -y_i, \forall i$. This vector will be updated every time that a Lagrange multiplier changes their value, because f_i also changes.
- α_i, are the Lagrange multipliers, with $i = 1, ...N$, associated to each training sample. All α_i receive an initial value near to zero: $2 * 10^{-16}$.

Algo 1 is the main procedure that evaluates all the samples and their associated Lagrange multipliers α_i. This is the main procedure that calls the *examineExample()* routine which optimizes and updates the Lagrange multipliers. The routine converges when the values of the multipliers did not

Algorithm 1: Main Routine

```
examineAll = 1
numChanged = 0
while examineAll OR numChanged ¿ 0 do
    if examineAll then
        forall the element i of the training set do
            numChanged = numChanged + examineExample(i)
        examineAll = 0
    else
        forall the element i of the training set where 0 < α_i < C do
            numChanged = numChanged + examineExample(i)
        examineAll = 1
```

change in a whole iteration. The output of the procedure is the list of Lagrange multipliers α_i and the bias b, which in combination with the training samples $\mathbf{x}_i$, and the targets y_i, any incoming test sample can be evaluated using equation 8.

The *examineExample* routine (algorithm 2), iteratively chooses all samples from the training set and search another sample to update their Lagrange multipliers. At the end of the cycle, if the values of the multipliers converged, the algorithm stops.

Algorithm 2: examineExample()

```
input  : i index sample
output: flagChanges
r_i = E_i · y_i
if (r_i < -tol AND α_i < C) AND (r_i > tol AND α_i > 0) then
    if number of non-zero AND non-C alphas > 1 then
        j = secondChoiceHeuristic(i)
        if takeStep(j,i) then
            return 1
    forall the examples j with 0 < α_j < C do
        j = randomly picked example
        if takeStep(j,i) then
            return 1
    forall the possible indexes j on train dataset do
        if takeStep(j,i) then
            return 1
return 0
```

The heuristic to choose on alg. 2 the second training sample, Platt calls it as *Second Heuristic*, seeks to maximize the numerator of eq. 25. Thus, the heuristic seeks to find the pair of samples where the difference in the classification error is important, and there is still place for improvements conditioned to the Lagrange multipliers.

The algorithm 4 is the function *takeStep*, which updates the values of the Lagrange multipliers with index i and j. If the change between new and old values is considerable, the bias threshold vector b and the error list E are updated, using alg. 5 and alg. 6 respectively. If the change is not noticeable, the function outputs a zero value (false) indicating this situation.

The algorithm 7 evaluates the objective function using the new value of α_2 and clip them inside the constrains, as showed on eq. 26.

The output of the SMO learning algorithm are the values of the α_i, and the bias b. In order to test a new input sample using eq. 8, these values, but also, the feature vectors $\mathbf{x}_i$ and the labels y_i of training samples.

Next section implements this algorithm to train a pool of classifiers which will recognize license plate numbers.

4 Multi-class SVM Recognition Framework

The License Plate Number recognition is considered as a $M = 10$ multi-class problem. This problem is tacked using the One-Against-All approach [10, 5].

Algorithm 3: secondChoiceHeuristic()

```
input  : E,i
output: j
sE,sidx = sort values of error vector E
if E_i > 0 then
    choose the sample with lowest error to maximize the step size |E_i − E_j|
    if sidx[1] equal i then
        j = sidx[2]
    else
        j = sidx[1]
else
    choose the sample with highest error to maximize the step size |E_i − E_j|
    if sidx[last] equal i then
        j = sidx[last-1]
    else
        j = sidx[last]
return j
end
```

4.1 One-Against-All Recognition Approach

The training dataset is composed of N samples: $\{x_1, y_1\}, ..., \{x_N, y_N\}$, where $x_i \in \Re^d$ is the input vector of concatenated HOE features, and $y_i \in \{0, 1, ..., M-1\}$ is the corresponding label.

The One-againt-all approach trains M binary SVM classifiers. To obtain a classifier for class i, the methodology gives a positive label to ith samples and a negative one to the rest of the training set. Equation 8 then becomes:

$$f_i(\mathbf{x}_t) = \sum_{j=1}^{N} \mathbf{1}_{y_j=i} \alpha_{i,j} K(\mathbf{x}_t, \mathbf{x}_j) - b_i \tag{28}$$

, where the notation $\mathbf{1}_{y_j=i}$ denotes an label function of y_j, which takes value 1 when the sample has the same label as the training class, $y_j = i$, and -1 otherwise, the variables $\{\alpha_{i,j}, b_i\}$ correspond to the output of the training algorithm taking the ith class as positive, as explained on section 3.3.

At the testing phase, a sample $\mathbf{x}_t$ is classified as in class i^* whose f_i^* produces the largest value of the svm outputs function of equation 28:

$$i^* = argmax_{i=1,...,M} f_i(\mathbf{x}_t) \tag{29}$$

4.2 Reliability Measures for Multi-Class SVM

The outputs of the multi-class recognition framework are analyzed in order to obtain a reliability measure allowing to accept or reject the classification. In [4] was implemented a strategy inspired on cognitive confidence values, proposed by Thome et al. [12].

The methodology defines two confidence variables: c_d and c_r. c_r gets the value from the largest SVM classification output for the $\mathbf{x}_t$ input sample. It scores the recognition performance, as how well the character is identified. The other variable c_d is associated to a discriminant performance, considering how well a classifier output is discriminant with respect to its $M-1$ competitors.

Algorithm 4: takeStep()

input : j,i
output: $flagChanged$
$s = y_i y_j$
if y_j *equals* y_i *(s=1)* **then**
  L=max(0,$\alpha_i + \alpha_j - C$)
  H=min(C,$\alpha_i + \alpha_j$)
else
  L=max(0,$\alpha_i - \alpha_j$)
  H=min(C,$C + \alpha_i - \alpha_j$)
if *If the boundaries overlap (L=H)* **then**
  return 0
$\eta = \mathbf{K}(j,j) + \mathbf{K}(i,i) - 2\mathbf{K}(j,i)$
if η **then**
  $a_2 = \alpha_i + y_i \cdot \frac{(E_j - E_i)}{\eta}$ **if** $a_2 < L$ **then**
    $a_2 = \mathrm{L}$
  else if $a_2 > H$ **then**
    $a_2 = \mathrm{H}$
else
  L_{obj}, H_{obj}= evaluateObjectiveFunction(j,i)
  if $L_{obj} < H_{obj}$ - *eps* **then**
    $a_2 = \mathrm{L}$
  else if $L_{obj} > H_{obj}$ + *eps* **then**
    $a_2 = \mathrm{H}$
  else
    $a_2 = \alpha_i$
if $|a_2 - \alpha_i| < eps \cdot (a_2 + \alpha_i + eps)$ **then**
  return 0
$a_1 = \alpha_1 + s \cdot (\alpha_i - a_2)$
updateThershold(j,i,a_1,a_2)
$\alpha_j = a_1$
$\alpha_i = a_2$
updateErrorList()
return 1

Algorithm 5: updateThreshold()

input : j,i,a_1,a_2
output: b
$b_1 = E_j + y_j \cdot (a_1 - \alpha_j) \cdot \mathbf{K}(j,j) + y_i \cdot (a_2 - \alpha_i) \cdot \mathbf{K}(j,i) + b$
$b_2 = E_i + y_j \cdot (a_1 - \alpha_j) \cdot \mathbf{K}(j,i) + y_i \cdot (a_2 - \alpha_i) \cdot \mathbf{K}(i,i) + b$
$b = 0.5(b_1 + b_2)$

The main difference with [12], is that on [4] the individual output values of the SVM classification functions are projected to a Gaussian space vector.

Let be $\mu_{\mathbf{x}_t}$ as the mean value of the M $f_i(\mathbf{x}_t)$ SVM classification outputs, and $\sigma_{\mathbf{x}_t}$ their standard deviation. It is defined a vector $\mathbf{C}$ with M elements, where each ith is obtained as:

Algorithm 6: updateErrorList()

input :
output: E
forall the *element i of the training set* **do**
  $svmOutput = \sum_{j=1}^{N} y_j \alpha_j \mathbf{K}(i,j) - b$
  $E_i = svmOutput - y_i$

Algorithm 7: evaluateObjectiveFunction()

input : j,i,L,H
output: L_{obj},H_{obj}
$s = y_j y_i$
$f_j = y_j(E_j + b) - \alpha_j \mathbf{K}(j,j) - s\alpha_i \mathbf{K}(j,i)$
$f_i = y_i(E_i + b) - s\alpha_j \mathbf{K}(j,i) - \alpha_i \mathbf{K}(i,i)$
$L_1 = \alpha_j + s(\alpha_i - L)$
$H_1 = \alpha_j + s(\alpha_i - H)$
$L_{obj} = L_1 f_j + L f_i + 0.5 L_1^2 \mathbf{K}(i,i) + 0.5 L^2 \mathbf{K}(j,j) + sLL_1 \mathbf{K}(j,i)$
$H_{obj} = H_1 f_j + H f_j + 0.5 H_1^2 \mathbf{K}(i,i) + 0.5 H^2 \mathbf{K}(j,j) + sHH_1 \mathbf{K}(j,i)$
return L_{obj}, H_{obj}

$$\mathbf{C}(i) = \frac{(f_i(\mathbf{x}_t) - \mu_{\mathbf{x}_t})^2}{\sigma^2}$$

Now, if the i^* index defines the largest value of the SVM classification outputs (see eq. 29), c_d and c_r confidence scores are defined as:

$$\begin{aligned} c_r &= \mathbf{C}(i^*) \\ c_d &= \frac{\mathbf{C}(i^*)}{\sum_{i=1, i\neq i^*}^{M} \mathbf{C}(i)} \end{aligned}$$

The output of the Multi-class Recognition Framework receives a reliability measure:

$$r = \frac{c_r c_d}{T_{CR} T_{CD}} \tag{30}$$

Both threshold values, T_{CR} and T_{CD} are estimated from the training dataset in order to validate the 99 % of samples. If $r > 1.0$ the output corresponds a character with a high discrimination ratio, and can be trusted as a real number. If $r < 1.0$ the input can be a number with a style not present on the learning dataset, or an image that not match to a number.

5 Experiments and Results

5.1 License Plate Numbers Datasets

The paper proposes a SVM Framework to recognize ten classes: the ten numbers from multistyle license plates. Figure 3 present some examples of the four types of license plates The dataset is composed of license plate's numbers extracted Dlavnekov dataset (EEUU) [3, 2], *COMVIS_cardataset_v1*

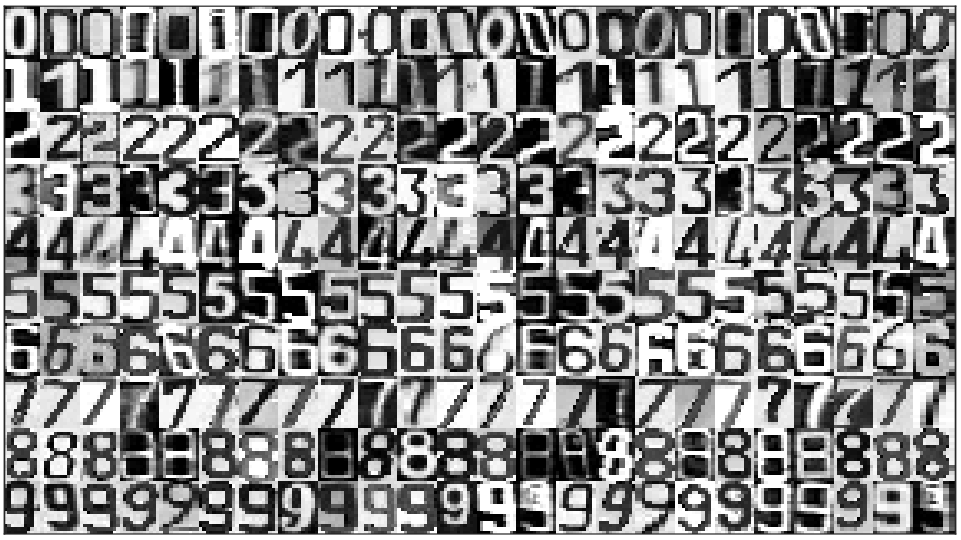

Figure 3: The figure shows the training and testing samples of the Multi-Style License Plate Numbers dataset, employed on the recognition system.

(Pakistan) [7], Medialab's dataset (Greece/European) [6], and an Argentinian license plate dataset. As can be seen, the datasets have opposite character and background colors.

Table 1 details the number of samples for each number class, discriminating between the Argentinean and the EEUU datasets.

Number	Argentine	EEUU	Greece	Pakistan	Total
'0'	33	14	49	19	115
'1'	33	22	29	15	99
'2'	42	21	85	30	178
'3'	32	36	74	37	179
'4'	36	61	69	37	203
'5'	32	47	95	21	195
'6'	26	25	75	20	146
'7'	26	25	74	17	142
'8'	30	22	62	35	149
'9'	28	23	74	14	139
Total	335	310	686	245	1545

Table 1: This Table shows the composition of the License Plate Numbers Dataset.

The training base will be composed for a randomly picked set within the four datasets, of $NTRAIN = 20$ samples of each number. Thus, the total number of training samples is $N = 200$. The remaining samples are employed for tests.

5.2 Resultats

Table 2 presents the results of applying license plate numbre recognition framework on the samples of the four datasets which were not be used on the training. As can be seen, the number were recognized with a high ratio, over 98 %. The overall performance of the system is evaluated by computing the mean of the diagonal, which gives 98.4 %.

	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'
'0'	94	1	0	0	0	0	0	0	0	0
'1'	0	78	0	0	0	0	0	0	0	0
'2'	0	0	159	0	0	0	0	0	0	0
'3'	0	1	0	157	0	0	1	0	0	2
'4'	1	0	0	0	181	0	0	0	0	0
'5'	0	0	0	0	0	170	5	0	0	0
'6'	0	0	0	0	0	0	125	0	1	0
'7'	1	4	0	0	0	0	0	116	0	0
'8'	0	0	0	0	0	0	0	0	126	0
'9'	1	0	0	1	0	0	0	0	0	114

Table 2: Confusion matrix of emotion License Plate Number recognition on the test samples from the four datasets.

The reliability measure, using eq. 30, is computed on each test sample. The percentage of samples with $r > 1.0$ is 74.27 %. This value can be improved by incrementing the size of the learning dataset incrementing the variable N.

References

[1] N. Dalal and B.Triggs, *Histograms of oriented gradients for human detection*, in IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, June 2005, pp. 886–893. http://dx.doi.org/10.1109/CVPR.2005.177.

[2] L. Dlagnekov and S. Belongie, *Recognizing cars*, Tech. Report CS2005-083, UCSD CSE, 2005.

[3] L. Dlagnekov and S. Belongie, *Ucsd/calit2 car license plate, make and model database*, 2005. http://vision.ucsd.edu/belongie-grp/research/carRec/car_rec.html.

[4] F. Gómez Fernández, P. Negri, M. Mejail, and J. Jacobo, *A multi-style license plate recognition system based on tree of shapes for character segmentation*, in Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, vol. 7042 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 443–450. http://dx.doi.org/10.1007/978-3-642-25085-9_52.

[5] Y. Liu and Y.F. Zheng, *One-against-all multi-class svm classification using reliability measures*, in International Joint Conference on Neural Networks, vol. 2, 2005, pp. 849–854.

[6] National Technical University of Athenas Medialab. http://www.medialab.ntua.gr/research/LPRdatabase.html, last accesed on 2012.

[7] COMSATS Institute of Information Technology. http://comvis.ciitlahore.edu.pk/downloads/comvis_cardataset_v1.0.html.

[8] John Platt, *Sequential minimal optimization: A fast algorithm for training support vector machines*, tech. report, Microsoft Research, 1998.

[9] F. Porikli, *Integral histogram: A fast way to extract histograms in cartesian spaces*, in IEEE Conference on Computer Vision and Pattern Recognition, 2005, pp. 829–836. http://dx.doi.org/10.1109/CVPR.2005.188.

[10] R. Rifkin and A. Klautau, *In defense of one-vs-all classification*, Journal of Machine Learning Research, 5 (2004), pp. 101–141.

[11] B. Schölkopf and A. Smola, *Learning with Kernels. Suppor Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, 2002. ISBN 0262194759.

[12] N. Thome, A. Vacavant, L. Robinault, and S. Miguet, *A cognitive and video-based approach for multinational license plate recognition*, Machine Vision and Applications, 22 (2010), pp. 389–407. http://dx.doi.org/10.1007/s00138-010-0246-3.

[13] V. Vapnik, *The nature of Statistical Learning Theory*, Springer, 1995. ISBN 9780387987804.

[14] P. Viola and M. Jones, *Robust real-time face detection*, International Journal of Computer Vision, 57 (2004), pp. 137–154. http://dx.doi.org/10.1023/B:VISI.0000013087.49260.fb.