A Comparative Study of SVM Models for Learning Handwritten Arabic Characters

MAHMOUD ZENNAKI, MAMOUN MAMOUNI, KADDOUR SADOUNI
Computer Science Department
University of Science and Technology Med Boudiaf of Oran
PO 1505 EL M'naoeur, Bir el Djir, Oran
ALGERIA
{mzennaki,mamouni_mamoun@yahoo.fr}, kaddour.sadouni@univ-usto.dz

Abstract: In order to select the best SVM model for a specific machine learning task, a comparative study of SVM models is presented in this paper. We investigate the case of learning handwritten Arabic characters and we make use of tabu search metaheuristic in order to scan a large space of SVM models including multi-class scheme (one-against-one or one-against-all), SVM kernel function and kernel parameters. These parameters have a great influence on final performance of the classifier and also on computation time. This work has involved the creation of a complete offline system for learning handwritten Arabic characters, generating a corpus of 4840 Arabic characters in their different positions (beginning, middle, end and isolated). Based on some theoretical interpretations and simulation results, the effect of SVM model on prediction rate and CPU time is discussed.

Key-Words: Character recognition, handwritten Arabic character recognition, Support Vector Machines, model selection, tabu search.

1 Introduction

Handwriting recognition has been the subject of intense research over the past twenty years, even if research works on the Arabic script are fewer in comparison with other types of writing (e.g. Latin or Japanese). In addition, the Arabic script shows a complex morphology of characters. This problem leads to high inertia at various levels including the choice of relevant primitives describing the morphology of characters and the need for a robust modeling and an efficient learning method to take into account any morphological variations of the Arabic script.

Among the techniques used for the recognition of Arabic manuscripts, we find the support vector machines (SVM) based on statistical learning theory [1]. The SVM introduced in the early 90s [2], has been very successful in almost all areas where they have been applied particularly in the field of handwriting recognition and has overcome many other learning methods.

For this reason, several studies based on SVM have been successfully achieved. Among them we find in [3] a segmentation approach applied to Arabic handwritten characters, which allows rebuilding offline, a tracing path similar to the online case. This approach uses a semi-skeletonization technique for monitoring character paths, and then applies a SVM classifier in the classification phase. The experiments have reached interesting recognition rate in reduced CPU time. In [4] the author presents a complete offline system using the SPIKE Neural Network (SNN) and SVM. The rate of recognition he gets is 76% for SVM and 69% for the SNN.

There are other works based on SVM ([5], [6]), but all these works share a well-known problem when using this technique which is the selection of the best SVM model, or in other words, the choice of parameters called hyper-parameters leading to the best prediction rate in reduced time. Hyper-parameters generally include the regularization parameter C and specific kernel function parameters.

Moreover, recognition of Arabic characters leads to "high" multi-class learning problem; in this case classical schemes one-against-one and one-against-all are commonly used to extend SVM (which are basically bi-class) to multi-class contexts. But considering a set of binary SVM to handle multiclass data makes the model selection more difficult; the number of hyper-parameters depends not only on kernel function, but also on classifiers dichotomies. This problem led us to propose an automatic selection technique based on tabu search metaheuristic rather than classical techniques like cross validation [7] and grid search [8] in order to achieve a comparative study of multi-class SVM

models. The use of metaheuristics is motivated by their ability to scan a large space of models and avoid to be trapped in bad local optima.

The rest of this paper is organized as follows. In section 2 we review support vector machines used for learning handwritten Arabic characters focusing on the main hyper-parameters used in SVM model selection. Section 3 is devoted to model selection techniques including the use of metaheuristics followed in section 4 by a description of a tabu search based algorithm for selecting SVM models. Then we describe in section 5 the recognition offline system we designed including primitive extraction and corpus building. Section 6 is intended for the comparative study based on simulation results and some theoretical interpretations. Finally the summary and conclusions of this work with some future lines of research are presented in the last section.

2 Support Vector Machines

Kernel Methods and particularly Support Vector Machines (SVM) [2], introduced during the last decade in the context of statistical learning, have been successfully used for the solution of a large class of supervised machine learning tasks such as categorization, prediction, novelty detection, ranking and clustering.

2.1 Basic notions

In general, the supervised learning problem can be stated as follows. Given a set of labeled training data or examples Z drawn from an unknown but fixed probability distribution, the task is to construct a decision rule that can predict future examples, called validation data, with high probability. The training set is $Z=\{(x_1,y_1), (x_2,y_2),\ldots,(x_n,y_n)\}$ where x_i represent the i^{th} example, and y_i its label or class. Using prior assumptions and by induction we can learn an estimated function which can be efficiently used to predict labels of future data (validation set). Many approaches have been used to estimate the decision rule such as neural networks, decision trees and more recently kernel methods and particularly the successful Support Vector Machines (SVM).

In their basic form SVMs are used in two-class supervised learning, where labels of examples are known to take only two values $y_i \in \{-1,+1\}$. Linear SVM finds a decision rule in the form of a hyperplane which maximizes the Euclidian distance to the closest training examples. This distance is called the margin δ, as depicted in figure 1.

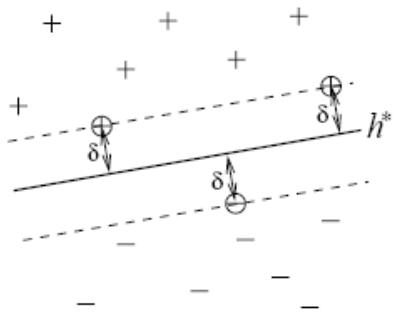

Fig. 1. Optimal plane maximizes margin.

SVM, in their general form, extend an optimal linear decision rule or hypothesis, in terms of an upper bound on the expected risk that can be interpreted as the geometrical margin, to non linear ones by making use of kernels k(.,.). Kernels represent dissimilarity measures of pairs of objects in the training set Z. In standard SVM formulations, the optimal hypothesis sought is of the form:

$$h(x) = \sum \alpha_i \, k(x,x_i) \quad (1)$$

where α_i are the components of the unique solution of a linearly constrained quadratic programming problem, whose size is equal to the number of training patterns. The solution vector obtained is generally sparse and the non zero α_i's are called Support Vectors (SV's). Clearly, the number of SV's determines the query time which is the time it takes to predict novel observations and subsequently, is critical for some real time applications.

It is worth noting that in contrast to connectionist methods such as neural networks, the examples need not have Euclidean or fixed-length representation when used in kernel methods. The training process is implicitly performed in a Reproducing Kernel Hilbert Space (RKHS) in which $k(x_i,x_j)$ is the inner product of the images of two examples x_i, x_j. Moreover, the optimal hypothesis can be expressed in terms of the kernel that can be defined for non Euclidean data such biological sequences, speech utterances etc. Popular positive kernels include (2, 3, 4, 5, 6):

Linear

$$k(x_i,x_j) = x_i.x_j \quad (2)$$

Polynomial

$$k(x_i,x_j) = (ax_i.x_j + b)^d \quad a > 0 \quad (3)$$

Gaussian

$$k(x_i,x_j) = \exp(-\|x_i - x_j\|^2/\sigma^2) \quad (4)$$

Laplacian

$$k(x_i,x_j) = \exp(-\gamma \, \|x_i - x_j\|) \quad \gamma > 0 \quad (5)$$

Sigmoïd

$$k(x_i,x_j) = \tanh(ax_i.x_j + b) \quad (6)$$

The notion of kernel functions confers to SVMs an important flexibility, a reasonable CPU time and an ability to generalize even in the presence of an important number of features. Particularly, the Gaussian kernel functions are called radial (Radial Basis Function or RBF) indicating that they depend on the distance between examples.

2.2 SVM formulation

Given training vectors $x_i \in IR^n$, i=1,…,m, in two classes, and a vector $y \in IR^m$ such that, $y_i \in \{1,-1\}$, Support Vector Classifiers solve the following linearly constrained convex quadratic programming problem:

$$\text{Maximize } \sum_i \alpha_i - 1/2 \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i,x_j)$$
$$\text{Subject to } 0 \le \alpha_i \le C \ \forall i \text{ and } \sum_i \alpha_i y_i = 0 \quad (7)$$

where C is a regularization parameter which influences classifier generalization when classes are strongly intertwined. The QP objective function involves the problem Gram matrix K whose entries are the similarities $k(x_i,x_j)$ between the patterns x_i and x_j. It is important to note, on one hand, that the pattern input dimension is implicit and does not affect to some extent the complexity of training, provided that the Gram matrix K can be efficiently computed for the learning task at hand. On the other hand, the patterns representation is not needed and only pair wise similarities between objects must be specified. This feature makes SVM very attractive for high input dimensional recognition problems and for the ones where patterns cannot be represented as fixed dimensional real vectors such as text, strings, DNA etc.

2.3 Multi-class extensions

Support Vector Machines are inherently binary classifiers and its efficient extension to multiclass problems is still an ongoing research issue ([9], [10], [11]). Several frameworks have been introduced to extend SVM to multiclass contexts and a detailed account of the literature is out of the scope of this paper.

Typically multiclass classifiers are built by combining several binary classifiers as depicted in figure 2. The earliest such method is the one-against-all (OVASVM) which constructs k classifiers, where k is the number of classes. The k^{th} classifier is trained by labeling all the examples in the k^{th} class as positive and the remainder as negative.

The final hypothesis is given by the formula:

$$h_{ova}(x) = \arg\max_{i=1,\ldots,k} (h_i(x)) \quad (8)$$

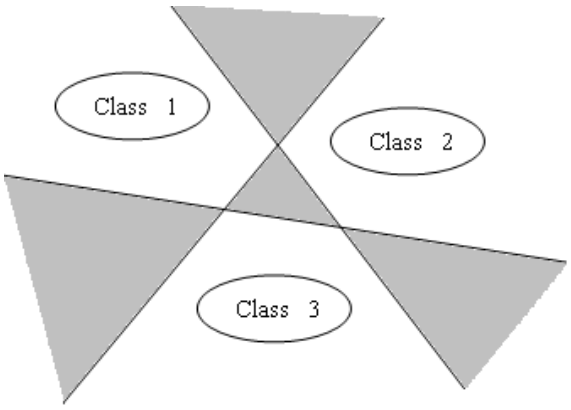

Fig. 2. Multi-class SVM

Another popular paradigm, called one-against-one, proceeds by training k(k-1)/2 binary classifiers corresponding to all pairs of classes. The hypothesis consists of choosing either the class with most votes (voting) or traversing a directed acyclic graph where each node represents a binary classifier (DAGSVM) [12]. There was debate on the efficiency of multiclass methods from statistical point of view. Clearly, voting and DAGSVM are cheaper to train in terms of memory and computer speed than OVASVM. Hsu and Lin [13] investigated the performance of several SVM multi-class paradigms and found that the one-against-one achieved slightly better results on some medium size benchmark datasets.

2.4 SVM model parameters

The major difficulty related to the use of SVM classifiers is the need to adjust variables conditioning the learning process. These variables are called hyper-parameters. The resolution of support vector machines problem involves the selection of several parameters; in addition to the regularization parameter C, we have:

- Parameter σ for RBF kernel
- Parameter γ for Laplacian kernel
- Parameters a,b,d for polynomial kernel
- Parameters a,b for sigmoid kernel.

Several methods of model selection are possible for selecting values of hyper-parameters. If the model uses a single hyper-parameter, one can try a finite number of values and choose the one that maximizes the prediction rate. This technique is however difficult to implement for two or more hyper-parameters. Furthermore, when we make use of one-against-one or one-against-all strategies in multi-class SVM, the number of hyper-parameters

increases considerably. For example, in one-against-all strategy, there are n×k variables to be optimized while in one-against-one strategy, we must consider n×k×(k-1)/2 variables, assuming that the same kernel function is used with each binary classifier (n represent the number of hyper-parameters and k is the number of classes).

3 SVM Model Selection Techniques

We present in this section the problem of hyper-parameters optimization as a model selection problem. We review the main techniques used in SVM model selection particularly the case of multi-class SVM.

3.1 Cross Validation

Probably the simplest and most widely used, cross-validation is a technique for testing a learned model, which can be achieved in several ways. The most common method is the k-Fold with k ∈ [4,10]. Given a training set Ap={X1,…,Xp} containing p elements, cross-validation can be achieved by the following five steps:

1. Cut all examples into k disjoint subsets of size p/k.
2. Learn the k-1 subsets.
3. Calculate the error on k^{th} part.
4. Repeat the process k times.
5. Obtain the final error by calculating the average of the k previous errors.

Cross-validation is simple to implement and use all data. It provides an estimate of the generalization error and can prevent over-fitting [7]. However, this technique yields meaningful results only if the validation and training sets are drawn from the same population.

3.2 Search Grid Procedure

It is a classical method which discretizes the models space. The limitation is the selection of models that use few parameters (≤ 2). Despite this defect, this method can draw surfaces corresponding to the evolving SVM capacities of generalization based on hyper-parameters values.

Many experiments in [8] show that the landscape of the generalization error through the use of this grid has many local minima, which shows that the only SVM hyper-parameters selection is itself a difficult problem.

These surfaces also show that the generalization error remains stable when changes on hyper-parameters are not important. The optimization of generalization does not require the search for precise values for these hyper-parameters. This allows an efficient search with relatively large discretization interval. For example, for a Gaussian kernel, we seek for the best couple (C, σ). So we will have exponential sequences $C=2^{-5}, 2^{-4},\ldots,2^{15}$ et $\sigma=2^{-15}, 2^{-14},\ldots 2^{3}$.

By using a heuristic we can avoid an exhaustive search of all parameters combinations even if an exhaustive search is still possible because the parameters are independent and we can easily parallelize the search [7].

3.3 Multi-class Model Selection

As we have already mentioned, the multi-class model selection is more complex than the binary case. Rather than fixing model parameters for all binary SVMs, two approaches have been proposed in the literature (see [14]) to deal with this problem, one called "local optimization" and the other called "global optimization".

The local approach we have used in this paper is to optimize each binary SVM classifiers for all combinations of classes. At the end, each binary SVM will have its own hyper-parameter values optimizing all margins corresponding to k (one-against-all strategy) or k×(k-1)/2 (one-against-one strategy) pairs of classes.

In contrast to this approach, the global optimization considers the set of all binary classifiers simultaneously. Despite its effectiveness, this approach is time consuming and cannot be used when dealing with large datasets.

3.4 Optimization by Meta-heuristics

The selection of optimal values of parameters; σ and C for a Gaussian kernel for example, consists to find an optimal model θ ≡ (C,σ). The problem of finding the model θ is a non-convex problem, so with several local minima particularly in the case of multi-class SVM. Among the proposed methods to solve this problem, a subset of them using metaheuristics is applied.

Metaheuristics are used in many areas of machine learning, for example in the problem of selecting a small subset of relevant attributes or relevant examples [15]. The advantage of metaheuristics methods is that they are generic and can solve a wide range of different problems, without need for deep changes in the algorithm used.

4 Tabu Search for Model Selection

4.1 Tabu Search Metaheuristic

Motivated by the encouraging results obtained in the solution of large scale of combinatorial optimization problems, Tabu Search (TS) metaheuristic were introduced in [16] and implemented later for NP-Hard problems [17] such as the notorious traveling salesman problem, vehicle routing problems etc.

Similar in spirit to many heuristics for solving hard combinatorial problems, TS explores the space of potential solutions by using a partial neighborhood. A short-term memory is used to prevent cycles regularly visiting the same local optimum. The algorithm keeps track of two lists: one contains candidate moves to be further explored, and the second one contains the most recent moves, called tabu, which are forbidden under some restrictions (tabu restrictions). In the case where a tabu move leads to a better solution, the restrictions on the move are lifted (aspiration criterion).

For more efficiency, intensification and diversification strategies are implemented in order to save computation time, both by speeding up the search in case of being far from a local optima (intensification), or by moving away the tabu search to different regions in order to avoid being trapped in local optima (diversification). The general scheme of tabu search can be as follows:

1. Generate an initial solution.
2. Create a candidate list of moves.
3. Select the best candidate, based on tabu restrictions and aspiration criterion (save the solution only if it is the best).
4. Stop criterion. If continue, change tabu restrictions and aspiration criteria, go to 2. Otherwise apply intensification and diversification strategies.

4.2 TS for SVM Model Selection

The experiments carried out in [18] show that model selection with tabu search is more efficient and faster than the search grid procedure. One advantage of using metaheuristics is that it is easy to extend to kernels with several parameters. In our recognition system we used the principle of tabu search to select kernel parameters. For example, for RBF kernel, the possible move correspond to add or subtract ΔC and $\Delta\sigma$ respectively to the regularization parameter C and the kernel width σ. So the parameters C and σ are reduced or increased depending on whether the move has increase or decrease the generalization capabilities. The procedure stops when the maximum number of iterations is reached.

Tabu search procedure for RBF kernel

1. Set tabu search parameters (tabu list size, number of iterations).
2. Select initial values for RBF kernel parameters (C and σ).
3. Add those values to tabu list.
4. Train and classify, T* = T (prediction rate).
5. Save parameters values and prediction rate in a file F.
6. Select the best move M which is non tabu among:
   M1 : C + ΔC and σ , M2 : C – ΔC and σ
   M3 : C and σ + $\Delta\sigma$, M4 : C and σ – $\Delta\sigma$
7. Add parameters value to tabu list.
8. Train and classify
   if T > T* then T* = T.
9. Add parameters values and T* to the file F.
10. Update the tabu list.
11. If the number of iterations is reached stop, otherwise go to 6.

We should mention here that the tabu list is managed as a FIFO list; the oldest tabu move is removed from the list and become non tabu.

This tabu search algorithm is embedded in a multi-class optimization process. We use for this the local optimization approach described in [14] which consists of optimizing each binary SVM classifiers for all combinations of classes such that each binary SVM is configured with its own hyper-parameters.

5 Recognition System of Handwritten Arabic Characters

This work has involved the creation of a complete offline system for learning handwritten Arabic characters, generating a database of 4840 Arabic characters. In this section we review first the corpus generation then we describe the main components of the system focusing on primitive extraction and detection of diacritical points.

5.1 Corpus generation

When evaluating the performance of a recognition system it is necessary to dispose of a consistent corpus. We have built in our laboratory (SIMPA) with the contribution of several researchers and students, a database containing 4840 examples of Arabic handwritten letters in different positions: isolated, beginning, middle and end (Figure 3). For

isolated letters, we have 100 x 28 = 2800 examples and for the others there are 30 x 68 = 2040 examples. The number of classes is 28+68 = 96. This corpus is divided into two datasets: training dataset (3840 examples) and validation dataset (1000 examples).

Fig. 3. Samples of our generated corpus

5.2 Description of the recognition system

5.2.1 Pretreatment

We performed in this phase the following operations:

- Binarization and other techniques to eliminate noise by thresholding.
- Expansion, done occasionally to enlarge the thickness plot (we obtain very small thickness while writing by graphics tablet).
- Erosion, used to remove noise (distant pixels) and extract only the part that form the character.
- Recovery operations of the picture by rotations to straighten the characters.

5.2.2 Normalization

Normalization is done in order to eliminate the size difference of characters which can affect the final results. However it may introduce a distortion of the original shape of the image which negatively affects recognition rates. Thus, the size of normalization should be selected carefully. The framework proposed in our system for the standardization of characters, is of size (70 x 70) which is high enough to prevent information loss [6].

5.2.3 Extraction of primitives

This phase is one of the most delicate and important in OCR. The recognition of a character begins with an analysis of its shape and extraction of its features (primitives) that will be used for identification. In our system, the extraction of primitives involves two steps: construction of the distribution matrix and detection of the diacritical point.

Distribution matrix

The construction of distribution matrix is an important part of our system. For a distribution matrix M of size N, the principle is to split the picture of the letter in NxN frames [i] [j], then count the number of black pixels in each frame and assign that number to the cell M[i][j] of the distribution matrix. We consider in figure 4 the distribution matrix 5x5 of the letter 'jim', written in Arabic 'ج' in its isolated form.

The different models of 'jim' must be as different as possible to cover the widest range of 'jim', but each one will still be closer to the class of 'jim' rather than any other class of letters. For the proposed system we used a distribution matrix of size 7 x 7 while the character picture is of size 70 x 70 after normalization. The matrix size is determined on the basis of experiments and represents an acceptable compromise.

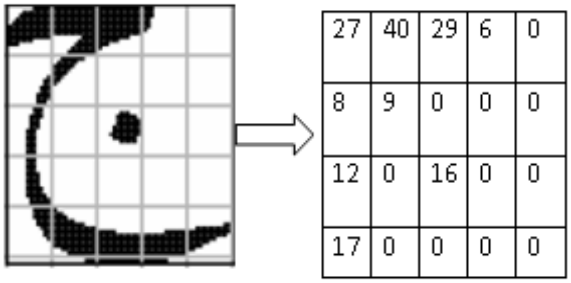

Fig. 4. Distribution matrix 5x5 of the letter “jim”

Detection of diacritical point

In Arabic, the existence of diacritical points is crucial. There are more than half of Arabic characters that have such point that is in many cases the only difference between two characters (two classes). That is why we have introduced in this phase of our algorithm, the detection of the diacritical point in the character and then add this property to the feature vector.

It is clear that in the case of characters which have similar shapes with the only difference being the position of diacritical point like (ب) (ت) (ـب), the distribution matrix can differentiate these characters since the entire shape is represented in this matrix. Detection of diacritics points is an additional step to emphasize this feature.

5.2.4 Building corpus

To allow direct use of multi-class SVM, the corpus is built so that each character corresponds to a line (or feature vector) in the corpus structured as depicted in figure 5.

0	1	2	3	…	48	49	50	…	58	59
5	43	60	60	…	65	60	100	…	100	100

Fig. 5. Corpus structure

The first value corresponds to the class of the Arabic character, followed by the values of the distribution matrix and the last ten values reflect the diacritical point. It is represented by the value VP repeated ten times and determined as follows. VP=100 if there is a diacritical point in the character, 0 otherwise. This value is repeated 10 times in order to give more consideration to this characteristic.

6 Experiments and Results

Our goal is to perform a series of experiments to select the best SVM model consisting of the type of classification one-against-one or one-against-all, the kernel function and kernel parameters, that give the best results in terms of prediction rate and CPU time. To achieve this goal, a comparative study is done between kernel function hyper-parameters for both multi-class learning schemes one-against-one and one-against-all.

First we split our corpus in two sets, one for training containing 3840 pictures (training dataset) and the other for test containing 1000 images (test dataset). For the model selection we have implemented our search strategy based on tabu search by varying and testing the SVM hyper-parameters for each binary SVM used in the multi-class learning. We begin by presenting the experiment results and we discuss these results based on some theoretical interpretations about the importance of the regularization parameter C, the number of support vectors, the initialization of kernel parameters as well as the problem of over-fitting. We also show that the optimization of SVM parameters leads not only to minimize the classification error, but also reduce the classifier complexity through the number of support vectors.

6.1 SVM one-against-all

We present here the experimental results using the type of SVM one-against-all. For each kernel, the corresponding parameters are considered and the last line of each table represents the best result that we found for each kernel.

Table 1. polynomial kernel one-vs-all results

Param. C	Param. d	coef	Training CPU time (s)	Test CPU time (s)	Prediction rate
1000	3	0	12,45	1,46	95,3 %
1000	10	10	11,46	1,12	95,5 %
1	3	0	12,71	1,39	96,0 %
1	3	1	12,20	1,18	96,4 %

Table 2. Gaussian kernel one-vs-all results

Param. C	Param. σ	Training CPU time (s)	Test CPU time (s)	Prediction rate
1000	1,92 E-05	8,82	1,14	96,6 %
100	1,92 E-05	9,50	1,14	96,9 %
100	1,83 E-05	9,73	1,12	97,0 %

Table 3. Laplacian kernel one-vs-all results

Param. C	Param. σ	Training CPU time (s)	Test CPU time (s)	Prediction rate
1000	1,92 E-05	16,09	1,68	95,9 %
100	1,02 E-07	12,98	1,57	96,4 %
1000	1,02 E-07	12,93	1,57	96,6 %

Table 4. Sigmoïd kernel one-vs-all results

Param C	Param. σ	coef	Training CPU time (s)	Test CPU time (s)	Prediction rate
1000	1,92 E-05	-10	13,21	2,15	58,5 %
1000	1,92 E-06	0	13,17	1,29	93,7 %
100	1,92 E-06	0	11,21	1,46	94,1 %
307	1,68 E-06	0	13,34	1,15	94,7 %

Table 5. Linear kernel one-vs-all results

Param. C	Training CPU time (s)	Test CPU time (s)	Prediction rate
5	10,81	0,78	90,8 %
1	11,84	0,76	91,0 %

6.2 SVM one-against-one

We present here experimental results using the type of SVM one-against-one

Table 6. Polynomial kernel one-vs-one results

Param. C	Param. d	coef	Training CPU time (s)	Test CPU time (s)	Prediction rate
1000	10	0	23,59	1,48	91,7 %
10	3	0	23,37	1,50	95,2 %
1000	3	10	22,93	1,53	97,0 %

Table 7. Gaussian kernel one-vs-one results

Param. C	Param. σ	Training CPU time (s)	Test CPU time (s)	Prediction rate
1000	1,02 E-05	22,87	1,43	97,1 %
10	1,92 E-05	24,50	1,50	97,4 %
1000	1,83 E-05	23,78	1.48	97,9 %

Table 8. Laplacian kernel one-vs-one results

Param. C	Param. σ	Training CPU time (s)	Test CPU time (s)	Prediction rate
10	1,92 E-05	28,28	1,82	95,0 %
1000	1,02 E-05	27,39	1,79	96,2 %

1000	1,92 E-05	28,40	1,85	96,3 %

Table 9. Sigmoïd kernel one-vs-one results

Param C	Param.	coef	Training CPU time (s)	Test CPU time (s)	Prediction rate
1000	1,92 E-05	1	22,23	1,82	58,6 %
1000	1,92 E-05	0	22,75	1,71	65,2 %
100	1,92 E-05	0	22,42	1,84	68,4 %
1000	1,00 E-06	0	22,81	1,54	97,2 %

Table 10. Linear kernel one-vs-one results

Param. C	Training CPU time (s)	Test CPU time (s)	Prediction rate
10	22,23	1,28	96,9 %
1000	22,14	1,34	96,9 %

6.3 Discussion

The experiments show the effectiveness of our system and particularly the effectiveness of our strategy for selecting hyper-parameters values based on tabu search by scanning a large area of parameters value. This choice has a great influence on the performance of the final classifier, and also on computation time.

We can notice first that the value of the regularization parameter C does not have a large influence on the results, though this parameter is quite critical for other problems where some of the data is not linearly separable even with using kernel functions.

The representation of letters we detailed in section 5.2.4. appears to be efficient. Indeed, even if the corpus contain some letters with many similarities like 'ج', 'ح' and 'خ', the prediction rates we obtained are very satisfying. The additional representation of diacritical points associated with data of distribution matrix has reduced considerably the misclassification rate.

As expected, SVM one-against-one strategy leads to better results than SVM one-against-all even if the CPU time is greater. We also found that the RBF kernel is best suited to the recognition of Arabic manuscripts. Indeed, this kernel gave better results than other kernels and has a recognition rate of 97.0% (for SVM one-against-all, σ=1.83E-05) and a recognition rate equal to 97.9% (for the SVM one-against-one, σ=1.83E-05).

We also notice that linear kernel gives significant results in one-against-one strategy. This kernel is stable, while sigmoid kernel is sensitive to any change of the parameter σ and requires an extensive search for good results. For the polynomial kernel, the parameter coef must be different from zero for best results, and has no importance for the sigmoid kernel. Laplacian kernel gives similar results between the two approaches one-against-one and one-against-all.

It is interesting to notice the improvement of prediction rates compared to those obtained with manual selection of SVM model. This improvement is due to the efficiency of automatic selection based on the local approach combined with tabu search metaheuristic to optimize the parameters of each binary SVM instead of assigning the same value for all binary SVM used in multi-class learning.

Another point is to emphasize and concerns the importance of kernel parameter initialization. Indeed, inadequate values often lead to over-fitting as outlined in [14], or may impede the convergence of SVMs due to a reduced ability of the classifier, particularly when the value of regularization parameter C is set to 1000 (when misclassifications are strongly penalized).

Finally, we note through numerous experiments performed that optimal selection of SVM models leads not only to minimize classification error, but also reduces the classifier complexity through the number of support vectors. The following table shows that the total number of support vectors obtained with automatic selection is about three times smaller than that obtained with manual selection.

Table 11. Comparison of total number of SVs

Kernel	Manual selection	Automatic selection
Polynomial	2432	1076
RBF	1621	679
Laplacian	1799	788
Sigmoïd	1832	845
Linear	2945	1311

7 Concluding Remarks

SVM is one of the machine learning techniques that has the greatest impact on pattern recognition by providing a theoretical framework. In this paper, we propose a system for recognizing handwritten Arabic letters based on SVM. The system was tested on a corpus containing 4840 examples and has given very good results in terms of recognition rate; it shows the effectiveness of the method for the extraction of primitives and the strategy used for SVM multi-class model selection based on tabu search. The system allowed us to make a comparative study of different SVM models used in the recognition of Arabic characters.

Future extensions are possible:

- Use a corpus generated from the segmentation of Arabic words datasets.

- Check other types of features to improve the recognition rate.

Finally, note the need to have a common protocol for validation of results between different approaches in recognition of handwritten Arabic script.

References:

[1] Vapnik V. N., "Statistical learning theory". Wiley, New York, 1998.

[2] Boser B., Guyon I., and Vapnik V., "A training algorithm for optimal margin classifiers". In *Fifth Annual Workshop on Computational Learning Theory*, Pittsburg,1992.

[3] Zaïz F., "SVM pour la reconnaissance de caractères manuscrits arabes", LESIA Laboratory, Computer Science Department, University Mohamed Khider Biskra, Algeria, 2010.

[4] Kadri M., "Méthode de reconnaissance de l'écriture arabe manuscrite en utilisant les réseaux neuronaux", Master Thesis, USTO MB, Oran, Algeria, 2010.

[5] Bouslimi R., "Système de reconnaissance hors-ligne des mots manuscrits arabes pour multi-scripteurs", Master thesis, USTOMB, Oran, Algeria, 2006.

[6] Lebrun G., "Sélection de modèles pour la classification supervisée avec des SVM (Séparateurs à Vaste Marge). Application au traitement et analyse d'images". Phd Thesis, University of Caen/Basse-Normandie, 2006.

[7] Hsu C.W., Chang C.C., and Lin C.J., "A practical guide to support vector classification". Technical Report, National Taiwan University, 2009.

[8] Lee J. and Lin C., "Automatic Model Selection for Support Vector Machines", Technical Report. csie.ntu.edu.tw/˜cjlin/papers/modelselect.ps.gz, 2000.

[9] Friedman J. H., "Another approach to Polychotomous classification". Technical report, Department of Statistics, Stanford University, 1996.

[10] Knerr S., Personnaz L., and Dreyfus G.. "Single-layer learning revisited: a stepwise procedure for building and training a neural network". In *Neuro-computing: Algorithms, Architectures and Applications*, J. Fogelman, editor, Springer-Verlag, 1990.

[11] Kreßel U. H. G., "Pairwise classification and support vector machines". In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods Support Vector Learning*, pages 255-268, The MIT Press, Cambridge, 1999.

[12] Platt J., Cristianini N., and J. Shawe-Taylor. "Large margin DAGs for multi-class classification". *Advances in Neural Information Processing System*, 12, MIT Press. 2000.

[13] Hsu C. W. and Lin C. J., "A comparison of methods for multi-class support vector machines", *IEEE Transactions on Neural Networks*, Vol. 13 415–425, 2002.

[14] Ayat N.E., Cheriet M., and Suen C.Y., "Optimization of the SVM kernels using an empirical error minimization scheme". In S.W. Lee and A. Verri, editors, *Pattern Recognition with Support Vector Machines*, volume 2388 of *LNCS*, pages 354–369. Berlin Heidelberg, July 2002.

[15] Dréo J., Petrowski A., Siarry P., and Taillard E., "Métaheuristiques pour l'optimisation difficile". Eyrolles Group, 2003.

[16] Glover F., "Tabu search: part I". In *On Computing*, 1(3), pages 190–206, 1989.

[17] Glover F., "Tabu search: part II". In *On Computing*, 2(1), pages 4–32, 1989.

[18] Cawley G. C., "Model Selection for Support Vector Machines via Adaptive Step-Size Tabu Search". In *ICANNGA*, 2001.