International Journal of Computer Applications (0975 – 8887)
Volume 2 – No.7, June 2010

SVM : Reduction of Learning Time

Sid Ahmed MOSTEFAOUI
Department of Computer Science
University of Tiaret
ALGERIA

Lynda ZAOUI
Department of Computer Science
University USTO
Oran-ALGERIA

ABSTRACT
Training a support vector machine (SVM) leads to a quadratic optimization problem with bound constraints and one linear equality constraint. Despite the fact that this type of problem is well understood, there are many issues to be considered in designing an SVM learner. In particular, for large learning tasks with many training examples, off-the-shelf optimization techniques for general quadratic programs quickly become intractable in their memory and time requirements. Here we propose an algorithm which aims at reducing the learning time, this algorithm is based on the decomposition method proposed by Osuna dedicated to optimizing SVMs: it divides the original optimization problem into sub problems computable by the machine in terms of CPU time and memory storage, the obtained solution is in practice more parsimonious_than that found by the approach of Osuna in terms of learning time quality, while offering similar performances.

Keywords;Classification;Learning;Support Vector Machines (SVM); Qquadratic optimization; Decomposition

1. INTRODUCTION
Support Vector Machines (SVMs) are a method which was introduced by [1]. This classification shows good performance in solving various problems such as pattern recognition or classification of texts. It is particularly well suited to handle high-dimension data such as text and images. An SVM is a learning algorithm allowing to learn a separator. This brings back the question of defining what a separator is. Give us a finite set of vectors of Rn, separated into two classes. Belonging to a group or another is defined by a label associated with each vector, which is inscribed "Class 1" or "Class 2". Finding a separator means building a function that takes a vector of our set, and can tell what group it is. SVMs are a solution to this problem, as would be a simple learning by heart of the classes associated with the vectors of our set. Theoretically we encounter an infinite number of separators to distinguish between the two classes. The objective of the SVM method is to decide the best separator that maximizes the margin of separation, so we are in a situation to solve a constrained optimization whose size depends on the number of documents constituting our training corpus. Practically, the solution of such systems becomes difficult (long learning time) if not impossible. Therefore we resort to decomposition methods, as their name suggests, which decompose the original problem in many sub optimization problems computable by machine.

In this paper we present the theory of SVMs, detailing the quadratic optimization techniques adopted by this method, then highlight the decomposition algorithm of OSUNA, lastly with the objective of reducing the learning time, we describe the proposed algorithm and we give some results of experiments on different corpora. Finally, we will propose some suggestions for future work.

2. SUPPORT VECTOR MACHINES

2.1 linear classification
A classifier is called linear when it is possible to express the decision function by a linear function in x which will designate a vector of Rn. where n is the number of components of the vectors containing the data. We can, in general, express this function by:

$$f(x) = \langle w, x \rangle + b = \sum_{i=1}^{n} w_i x_i + b$$

Where w ∈ Rn and b ∈ R are parameters, and x ∈ Rn is a variable.

This classifier does not only give values -1 (class 1) or 1 (class 2), but we will say that when the result fw,b (x) is positive, the vector x belongs to the same class as the examples label 1, and when the result is negative, the vector x belongs to the same class as examples of label -1.

Note that equation fw,b(x) = 0 defines the boundary of separation between the two classes, and that this border is a hyper plane in the case of affine linear separator.

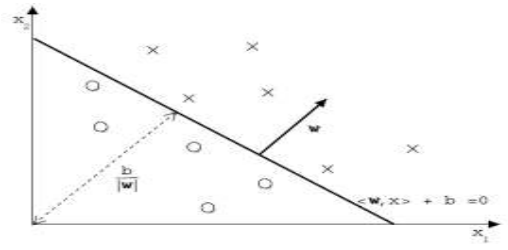

FIG.1 Representation, in R^2 , of the hyperplane corresponding to the decision function of a linear classifier

2.2 Margin
Let S= {(xi, yi) ∈ Rn x {-1,1}}, i=1…n, the training set with labels $y_i \in \{-1,1\}$, S+= {x/ (x,y) ∈ S and y=1} and S-= {x/ (x,y) ∈ S and y=-1}.

The geometric margin is the Euclidean distance taken perpendicularly between the hyper plane, characterized by w and b, and the example xi, this margin is defined as:

$$\Psi_{w,b}(x_i, y_i) = y_i \left(\frac{w}{\|w\|} x_i + \frac{b}{\|w\|} \right)$$

The margin of the training set compared to the hyper plane characterized by w and b is defined as:

$$\Psi_{w,b} = \min_{i=1 \ldots n} \Psi_{w,b}(x_i, y_i)$$

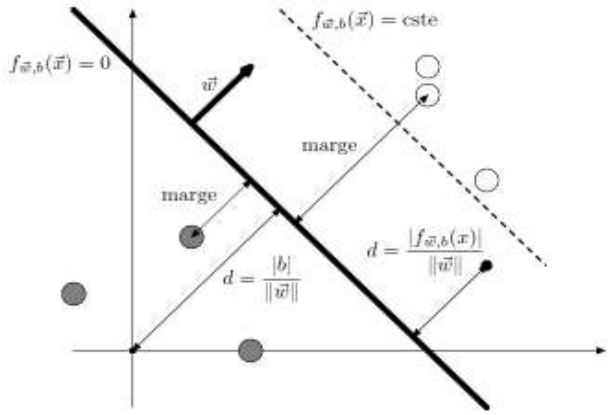

FIG.2 : Definition of a separator $f_{w,b}$

Classifiers with the objective of maximizing the margin of the training set are called maximum margin classifiers. SVM, in particular, is one of them.

2.3 The canonical hyperplane

For the objective of normalization, we can define two planes located on both sides of the hyperplane and parallel to it. Note that the examples closest are on the canonical hyperplanes, and are called support vectors. Figure (FIG.3.a) illustrates this situation.

It is therefore possible to resize w and b so that the two parallel planes have respectively the equation:

$$\langle w, x \rangle + b = 1 \quad \text{and} \quad \langle w, x \rangle + b = -1$$

These two hyper planes are called canonical hyper planes.

Note that the margin of the canonical hyper plane is 1/‖w‖.

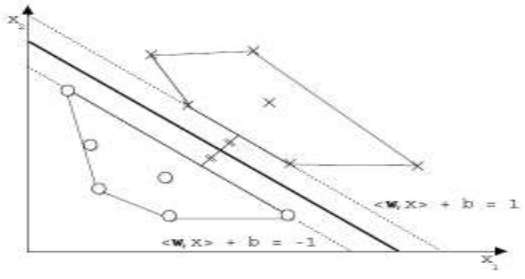

FIG.3.a : Canonical hyperplanes

For the separator with maximum margin, these examples have a margin larger than the examples with smallest margin of other possible separators.

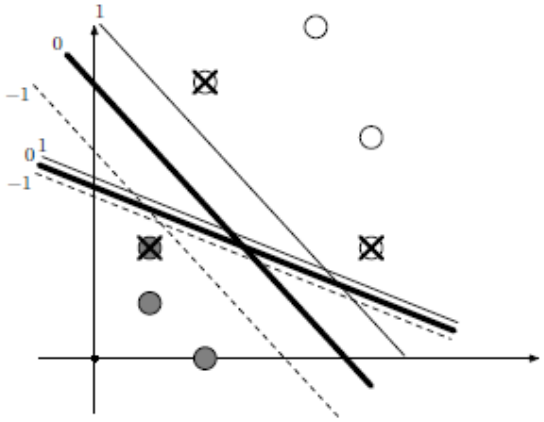

FIG.3.b: Infinity hyperplane separators, and the optimal hyperplane with maximum margin.

2.4 Optimization problem

The width of the band formed by the canonical hyper planes is 2/‖w‖. To find the maximum margin separator, it is sufficient to search among the separators checking for all the examples yi*f (xi) ≥ 1, the separator for which ‖w‖ is minimal.

Now we can formulate a mathematical optimization problem as its solution provides the optimal hyper plane (maximizing the margin):

$$\textbf{(QP1)} \ \textit{Minimize} \qquad W(w,b) = \frac{1}{2}\|x\|^2$$

$$\text{Subject to} \qquad y_i\left(\langle w, x_i \rangle + b\right) \geq 1$$

This is a quadratic optimization problem. Its objective function is the square of the inverse of the double marginalization. The only constraint states that the examples should be properly classified and they do not exceed the canonical hyper planes.

To avoid the problems of conventional methods of machine learning (over fitting, curve of dimensionality), it is necessary to introduce a dual formulation of the problem. For this objective, we must form what is called the Lagrangian. It is to enter the constraints in the objective function and weighted each one by a dual variable, which are called Lagrange multipliers.

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i \left(\langle w, x_i \rangle + b \right) - 1 \right] \qquad (1)$$

In determinant conditions [2] of our optimization problem (QP1), we find that:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \qquad w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

By substituting in (1) we can formulate the following dual problem:

$$(QP2) \quad Minimier \quad W(\alpha) = -\sum_{i=1}^{n} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

$$s.t \quad \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \alpha_i \geq 0 \qquad \forall i = 1...n \end{cases}$$

Now, we have all the necessary elements to express the decision function of our linear classifier:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle x, x_i \rangle + b$$

2.5 Soft Margin

In general, it is not possible to find a linear separator in the space of redescription. It is also possible that some examples are mislabeled and that the separating hyper plane is not the best solution to the problem of classification.

[3] propose a technique called soft margin, which tolerates bad rankings. The technique seeks a separating hyper plane that minimizes the number of errors with the introduction of slack variables ξk, which can relax the constraints on the vectors of learning:

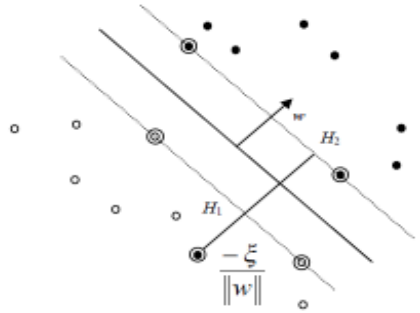

FIG.4 : The linear hyperplanes for classification problem nonlinearly separable.

As before, the Lagrangian of this problem is written:

$$(QP3) \quad Minimiser \quad L(w,b,x_i) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

$$s.t \quad \begin{cases} y_i\left(\langle w, x_i \rangle + b\right) \geq 1 - \xi_i \qquad \forall i = 1..n \\ \xi_i \geq 0 \qquad \forall i = 1..n \end{cases}$$

By forming the Lagrangian, then applying the theorem (KKT), we find the following dual form:

$$(QP4) \quad Maximiser \quad W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

$$s.t \quad \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C \qquad \forall i = 1...n \end{cases}$$

Where *C* is a parameter set in advance. The larger C is, the more the errors are penalized.

2.6 Nonlinear decision surfaces

If data are not linearly separable, a solution to better separate the examples is to project them into a different space (the feature space), and realize a linear separation in that space there.

We denote the feature space F, and the mapping Φ into this space, we have:

$$\Phi : X \rightarrow F$$

$$\Phi(x) = \begin{pmatrix} \varnothing_1(x) \\ \varnothing_2(x) \\ . \\ . \\ \varnothing_n(x) \end{pmatrix}$$

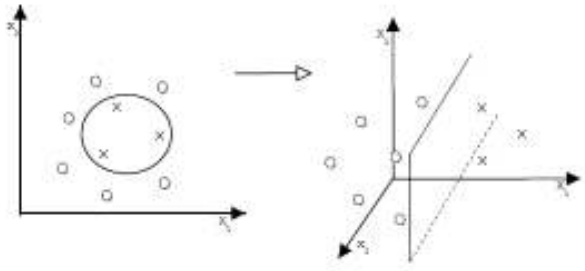

FIG.5 : A mapping Φ making examples linearly separable

We end up making the linear separation on the new corpus $\overline{S}$. This gives a separator, given by $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ and b.

In fact, we will not proceed like this, and we prefer to avoid the explicit calculation of $\Phi(x)$ by noting that the optimization problem posed in the previous section involves the vectors via dot products between them. Note k (x, z) the product $\langle \Phi(x), \Phi(z) \rangle$, where the function k(x, y) is a kernel function. Work on the corpus $\overline{S}$ returns to work on the initial corpus S with previous methods, but replacing all instances $\langle \bullet, \bullet \rangle$ by $k(\bullet, \bullet)$.

The trick is that we will not make this projection, because we calculate k (x, z) otherwise. In fact, k (x, z) is a function we will give, ensuring that there is good in theory Φ a projection in a space that does not seek to describe. Thus, we directly calculate k (x, z), whenever the previous algorithm requires a dot product, and that's all! The projection in the large feature space is implied.

Example of RBF kernel (Radial Basis Function)

The generic form of this kernel is:

$$k(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$$

Where the parameter σ adjusts the width of the Gaussian.

The vector w has a very clear geometric meaning.

2.7 Formulation of SVM

The problem to be solved by the SVM is a quadratic optimization problem under linear constraints, its version matrix is as follows:

$$(QP5) \qquad Minimiser \quad W(\alpha) = -\alpha^T 1 + \frac{1}{2}\alpha^T Q\alpha$$

$$S.t \quad \begin{cases} \alpha^T y = 0 \\ 0 \le \alpha \le C1 \qquad \forall i = 1...n \end{cases}$$

Where the matrix Q is defined by $Q_{ij} = y_i y_j k(x_i, x_j)$

3. THE DECOMPOSITION ALGORITHMS

The size of the optimization problem (QP5) depends on the number of training set examples noted l. The size of the matrix Q is l2, for a training set of 10,000 examples and more, it becomes impossible to keep Q in memory.

Many standard implementations of (QP5) solvers require explicit storage of Q which prohibits their application. To solve quadratic optimization systems of this size, the world of optimization proposed decomposition methods, which use subsets of the training set at each stage in order to solve small size problems.

3.1 The property used: parsimony

Note first that the forms witch will be optimised are convex and thus admit a single optimal solution. Note also that the constraints are linear and finite in number: the optimal solution α checks for each component α_i Karuch Kuhn Tucker (KKT) conditions,

ie: $\alpha_i = 0 \quad \rightarrow \quad y_i f(x_i) > 1$

$0 < \alpha_i < C \quad \rightarrow \quad y_i f(x_i) = 1$

$\alpha_i = C \quad \rightarrow \quad y_i f(x_i) < 1$

The general principle of decomposition methods based on the observation that only the unconstrained points in the solution requires the calculation of their coefficients: it is parsimony. Indeed, others have a fixed value for the coefficient, given by the problem. This has led to different techniques and different decomposition algorithms that we will explain.

3.2 Common structure solvers

The iterative algorithms for solving share a common structure. The methods differ in how they then distribute the points and the calculation of α.

General Decomposition Algorithm

1: initialization

2: While current solution is not optimal do

a: update the distribution of groups

b: calculate the coefficients α_i corresponding to changes

5: end while

3.3 The decomposition algorithm of Osuna

3.3.1 Optimality Conditions

The QP problem we have to solve is the following:

$$(QP6) \qquad Minimize \quad W(\alpha) = -\alpha^T 1 + \frac{1}{2}\alpha^T Q\alpha$$

$$S.t \quad \begin{cases} \alpha^T y = 0 \\ -\alpha \le 0 \quad \forall i = 1...n \\ \alpha - C \le 0 \end{cases}$$

So the conditions of Kuhn-Tuker are necessary and sufficient for optimality. The KT conditions are as follows:

$$\nabla W(\alpha) + \left(\lambda^{eq} y - \lambda^{lo} + \lambda^{up}\right) = 0$$

$$\forall i = 1..n \qquad \lambda_i^{lo}(-\alpha_i) = 0$$

$$\forall i = 1..n \qquad \lambda_i^{up}(\alpha_i - C) = 0$$

$$\alpha^T y = 0$$

$$\lambda^{up} \ge 0 \qquad (1)$$

$$\lambda^{lo} \ge 0 \qquad (2)$$

$$0 \le \alpha \le C1$$

In order to derive still other algebraic expressions optimality conditions (1) and (2) we assume the existence of some α_i such that $0 < \alpha_i < C$, and consider three possible values that each component α_i can have:

- **Case 1 :** $0 < \alpha_i < C$ for the first three equations of the KT conditions we have:

$$(Q\alpha)_i - 1 + \lambda^{eq} y_i = 0 \qquad (3)$$

Using the results of [3] and [4], we can easily prove that when α is strictly between 0 and *C,* the following equality:

$$y_i\left(\sum_{j=1}^{l}\alpha_j y_j k(x_i, x_j) + b\right) = 1 \qquad (4)$$

And we have $(Q\alpha)_i = y_i\left(\sum_{j=1}^{l}\alpha_j y_j k(x_i, x_j)\right)$

By combining this expression with (3) and (4), we obtain immediately: $\lambda^{eq} = b$

- **Case 2 :** $\alpha_i = C$

By definition: $f(x_i) = \left(\sum_{j=1}^{l}\alpha_j y_j k(x_i, x_j) + b\right)$

and noting that

$$(Q\alpha)_i = y_i\left(\sum_{j=1}^{l}\alpha_j y_j k(x_i, x_j)\right) = y_i\left(f(x_i) - b\right)$$

we conclude that $y_i f(x_i) \le 1$

Case 3 : $\alpha_i = 0$ By applying a similar algebraic manipulation as the one described for case 2, we obtain:

$$y_i f(x_i) \ge 1 \qquad (5)$$

1) ***Proposition :***

Given an optimal solution of a sub problem defined on B, the operation of replacing $\alpha_i = 0 \quad i \in B$, with $\alpha_j = 0 \quad j \in N$, j E N, satisfying $y_i f(x_i) < 1$ generates a new sub problem that when optimized, yields a strict improvement of the objective function [4].

2) ***The Decomposition algorithm***

Suppose we can define a fixed-size working set B, such that $|B| \le l$, and it is big enough to contain all support vectors , but small enough such that the computer can handle it and optimize it using some solver. Then the decomposition algorithm can be stated as follows:

- Arbitrarily choose $|B| \le l$ points from the data set.
- Solve the subproblem defined by the variables in *B*.
- While there exists some $j \in N$, such that $y_i f(x_i) < 1$, replace $\alpha_i = 0 \quad i \in B$, with $\alpha_j = 0 \quad j \in N$

and solve the new sub problem.

Notice that, according to the optimality conditions described above, this algorithm will strictly improve the objective function at each iteration and therefore will not cycle. Since the objective function is bounded W(α) is convex quadratic and the feasible region is bounded), the algorithm must converge to the global optimal solution in a finite number of iterations.[4]

4. PROPOSED ALGORITHM

4.1.1 Algorithm strategy

Our strategy is based on the same principle to the proposal of [4], we divide all variables into two sets B and N such that optimality conditions will hold in the subproblem defined only for variables in the set B, called working set, then we decompose the vector α into two vectors α_B, α_N and put $\alpha_N = 0$. The difference is in the replacement strategy elements of B satisfying $\alpha_i = 0$ by the elements of N which satisfy $y_j f(x_j) < 1$, where our method will select the closest elements to the current hyperplane in terms of functional margin. Applying the following algorithm guarantees that, in each iteration the objective function take a faster step to the minimum, than Osuna algorithm.

4.1.2 Algorithm

- Arbitrarily choose $|B| \le l$ points from the data set.
- Solve the subproblem defined by the variables in B.
- While there exists some $j \in N$, such that $y_i f(x_i) < 1$, replace $\alpha_i = 0 \quad i \in B$, with $\alpha_j = 0$

Where

$$j \in N \text{ and } \quad y_j f(x_j) < y_k f(x_k) \quad \forall k \in N$$

and solve the new subproblem.

3) ***Proof***

We assume the existence of α_p such as $0 < \alpha_p < C$ suppose also, without loss of generality that $y_p = y_j$ (the proof is analogous if $y_p = -y_j$) Then there exists some $\varepsilon > 0$ such that $\alpha_p - \delta > 0$ for $\delta \in (0, \varepsilon)$. Notice also that $f(x_p) = y_p$,

Now , consider $\overline{\alpha} = \alpha + \delta e_j - \delta e_p$ where e_j and e_p are the jth and pth unit vectors , and notice that the pivot operation can be handled implicitly by letting $\delta > 0$ and by holding $\alpha_i = 0$. The new cost function $W(\overline{\alpha})$ can be written as:

$$W(\overline{\alpha}) = -\overline{\alpha} \cdot 1 + \frac{1}{2}\overline{\alpha} \cdot Q\overline{\alpha}$$

$$= -\alpha \cdot 1 + \frac{1}{2}\overline{\alpha}\left[\alpha \cdot Q\alpha + 2\alpha \cdot Q(\delta e_j - \delta e_p) + (\delta e_j - \delta e_p) \cdot Q(\delta e_j - \delta e_p)\right]$$

$$= W(\alpha) + \delta\left[\frac{f(x_j) - b}{y_j} - 1 + \frac{b}{y_p}\right] + \frac{\delta^2}{2}\left[K(x_j, x_j) + K(x_p, x_p) - 2y_j y_p K(x_j, x_p)\right]$$

$$= W(\alpha) + \delta\left[y_j f(x_j) - 1\right] + \frac{\delta^2}{2}\left[K(x_j, x_j) + K(x_p, x_p) - 2y_j y_p K(x_j, x_p)\right]$$

Therefore, since $y_j g(x_j) < 1$, by choosing δ small enough we have $W(\overline{\alpha}) < W(\alpha)$.

Now, we assume the existence of an element α_k witch verifies $y_k g(x_k) < y_j g(x_j) < 1$, and consider $\overline{\overline{\alpha}} = \alpha + \delta e_k - \delta e_p$

Then we have

$$W(\overline{\overline{\alpha}}) - W(\overline{\alpha}) = \delta\left[(y_k f(x_{kj})) - (y_j f(x_j))\right] + \frac{\delta^2}{2}\left[(K(x_k, x_k) - 2y_k y_p K(x_k, x_p)) - (K(x_j, x_j) + 2y_j y_p K(x_j, x_p))\right]$$

To this effect, since $y_k g(x_k) < y_j g(x_j) < 1$, by choosing δ small enough we have

$$W(\overline{\overline{\alpha}}) < W(\overline{\alpha}) < W(\alpha)$$

4) Implementation

We chose to code this algorithm in the Matlab environment to take advantage of its optimizer quadprog () which can go well for a number of 1000 learning examples.

Our algorithm is designed to perform the classification of text documents. Our choice of representation has naturally focused on the vector model. In the case of the bag of words representation, the training set used is a sample of the database "Reuters-21578" available at**:** A second perspective is to use quadratic optimizers that rely on other techniques such as (Interior Point Method: IPM), which seem to give very good results in terms of computational time and accuracy of the solution.

http://download.joachims.org/svm_light/examples/example1.tar.gz, This set contains 1000 positive examples (class 1) and 1000 negative examples (class -1) in the file "train.dat. The file "test.dat" contains the database of test data where we have 300 positive examples (class 1) and 300 negative examples (class -1).

5) Results

The following tables show the results obtained by the two learning algorithms based on parameter values C=1000, δ =10 for RBF kernel and different samples taken from the learning set "Reuters-21578". We will always initialize the same working set for both algorithms.

	Time CPU (min)	***Test***
Osuna	5,01	96,99 %
Changed Ossuna	4,43	97,16 %

Tab.1 : Results of algorithms for 600 examples of learning

	Time CPU (min)	***Test***
Osuna	26,27	97,16 %
Changed Ossuna	15,88	97,16 %

Tab.1 : Results of algorithms for 800 examples of learning

6) Discussion

We started learning with a set of 600 examples for both algorithms including 300 positive and 300 negative, this number of examples will be increased for each experiment under the same parameters. The tables above show that, the learning time elapsed from the amended Osuna algorithm is always better than the time of the original algorithm whose difference is proportional to the size of the corpus. We also note that the model deduced always gives the same results on the test set about 97% for all experiments.

5. CONCLUSION AND PERSPECTIVES

The objective of our work is the development of a text classifier by learning using the method Support Vector Machine (SVM). This work deals with the learning and the formulation of principles of the SVM method, which becomes a quadratic optimization problem untold by the machine. In this context, several approaches are proposed and which are generally based on the principle of decomposition. we have discussed, in particular, the algorithm of Osuna. Based on the same principle of this algorithm, we have proposed a new way to define the sub-problems to optimize (working set). Finally, in the phase of implementation we have chosen a sample of the corpus "Reuters-21578 "containing 2000 learning examples and 600 test items, then we have applied both algorithms on several samples of the corpus by increasing the size of sample for each experiment. The results obtained show that our strategy has worked to reduce the learning time compared to the algorithm of Osuna, and that the model derived by each algorithm gave the same results on the classification of the test corpus.

One perspective in this work is the implementation of the algorithm using the technique of caching for the Hessian matrix Q, to exploit its influence on the training duration.

Bibliography

[1] V. Vapnik, «*The Nature of Statistical Learning Theory*». Springer Verlag, New York, 1995.

[2] W. Kuhn et A. W. Tucker. « *Nonlinear programming »*. In Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics, pages 481–492, Berkeley, 1951.University of California Press.

[3] C. Cortes et V. Vapnik. « *Support vector networks*». Machine Learning, 20:1-25, 1995.

[4] E. Osuna, R. Freund, and F. Girosi. «*Support vector machines: Training and applications*». A.I. Memo 1602, MIT A. I. Lab., 1997.