Heuristics to Reduce the Training Time of SVM Algorithm

Ariel García-Gamboa, Neil Hernández-Gress, Miguel González-Mendoza, Rodolfo Ibarra-Orozco, and Jaime Mora-Vargas
ITESM-CEM, Carretera Lago de Guadalupe Km 3.5, Atizapán de Zaragoza, Estado de México, C.P. 52926, México
{ariel.garcia, ngress, mgonza, rodolfo.ibarra, jmorag}@itesm.mx
(Paper received on September 06, 2006, accepted on September 27, 2006)

Abstract. This paper presents two initialization algorithms together with the study of the Quadratic optimization problem (QP), in order to reduce the training time of Support Vector Machines (SVMs). The QP problem is very resource consuming, because the quadratic form is dense and the memory requirements grow square the number of data points. The SVM-QP problem can be solved by several optimization strategies, but for large-scale applications, a general extension consisting in the decomposition of the QP problem into smaller ones is used. The support vectors found in the training of SVMs represent a small subgroup of the training patterns. The algorithms used in this approach are to initialize the SVMs making a fast approximation of the points standing for support vectors, and finally, making the training phase only with those data. The combination of these initialization algorithms and the decomposition approach, coupled with different QP solvers specially arranged for the SVM-QP problem are compared using some well-known benchmarks.

1. Introduction

Support Vector Machines (SVMs) is a well-known technique for training separating functions in pattern recognition tasks and for function estimation in regression problems. In several problems has shown its generalization capabilities. The mathematical problem and solutions were settled by Vapnik and Chervonenkis in [1]. Training a Support Vector Machine (SVM) consists in solving a Quadratic Programming (QP) problem. While solving this QP problem, SVMs solves two problems that classical Neural Networks have: 1) choose an optimal topology and 2) adjust the parameters of the network. Since the number of variables in the QP-problem is equal to the number of training patterns, the optimization problem becomes challenging, because the quadratic form is dense and the memory requirements grow square the number of data points. Both algorithms used in this paper, used to train a SVM, are based on the idea that the support vectors represents a small subgroup of the training patterns and if we train a SVM only with those data, the same results are obtained as trained using the entire data base.

2. Support Vector Machines

Support Vector Machines is a well-known technique for solving classification, regression and density estimation [1] problems. This learning technique provides a

Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 237-246

convergence to a globally optimal solution and, for several problems it has shown better generalization capabilities than other learning techniques.

In agreement with the inductive principle of the structural risk minimization, [1], in the statistical learning theory, a function, which describes correctly a training set X and which belongs to a set of functions with a low VC (Vapnik-Chervonenkis) dimension, will have a good generalization capacity independently of the input space dimension. Based on this principle, the Support Vector Machines [1] have a systematic approach to find a linear function, belonging to a set of functions with a low VC dimension.

The principal characteristic of SVMs for pattern recognition is to build an optimal separating hyperplane between two classes. If this is not possible, the second great property of this method resides in the projection of X space into a Hilbert space F of highest dimension, through a function $\varphi(x)$. One example is the internal product evaluated using kernel functions:

$$k(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)^{\mathrm{T}}\phi(\mathbf{x}_j),\ i,j = 1,...,l \tag{1}$$

satisfying Mercer conditions, like the Gaussian kernel:

$$k(\mathbf{x}_i,\mathbf{x}_j,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x_i - x_j)^2}{2\sigma^2}\right) \tag{2}$$

Thus, thanks to the freedom of using different types of kernels, the optimal separating hyperplane corresponds to different non-linear estimators in the original space.

For classification tasks, the main idea can be stated as follows: given a training data set $(\mathbf{X})$ characterized by patterns $x_i \in \Re^n, i = 1,...,n$ belonging two possible classes $y_i \in \{1,-1\}$, there exist a solution represented by the following optimization problem:

Maximize
α
$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) \tag{3}$$

Under the constraints
$$\sum_{i=1}^{l} y_i\alpha_i = 0, \qquad 0 < \alpha < c \tag{4}$$

where α_i are the Lagrange multipliers introduced to transform the original problem formulation with linear inequality constraints into the above representation, [1]. The parameter C controls the misclassification level on the training data and therefore the margin. The $k(x_i,x_j)$ term represents the so called kernel trick and is used to project data into a Hilbert space F of higher dimension using simple functions for the computation of dot products of the input patterns: $k(x_i,x_j) = \phi(x_i)^T\phi(x_j)$, $i,j = 1,...,n$. Once one has the solution, the decision function is defined as:

$$f(\mathbf{x}) = \mathrm{sign}\left(\sum_{i=1}^{l}\alpha_i y_i k(\mathbf{x}_i,\mathbf{x}) + b\right) \tag{5}$$

The solution to the problem formulated in (4) is a vector $\alpha_i^* \geq 0$ for which the α_i strictly greater than zero are the support vectors.

3. Quadratic programming optimization problem

Let us define the matrix Q as $Q_{ij} = y_i y_j k(x_i, x_j)$ $i, j = 1,...,l$ and the vectors $\alpha = [\alpha^1,...,\alpha^l]^T$, $1 = [1^1,...,1^l]^T$, $y = [y^1,...,y^l]^T$ and $C = [C^1,...,C^l]^T$, the SVM-QP can be written in matrix form as:

$$\underset{\alpha}{\text{Minimize}} \quad q(\alpha) = \frac{1}{2}\alpha^T Q\alpha - 1^T\alpha \tag{6}$$

$$\text{Under the constraints} \quad y^T\alpha = 0, \quad 0 \leq \alpha \leq C \tag{7}$$

The global minimum of α is guaranteed because of the problem's convexity.

3.1. Optimality conditions

The QP problem must satisfy first and second order conditions, but Ec (6-8) make any first order critical point a global solution, [8]. In order to solve the problem, the classical Lagrange duality theory is used and then we have: $\gamma \in \Re$ to (7) and $\beta = [\beta^1,...,\beta^l]^T$ and $\chi = [\chi^1,...,\chi^l]^T$ to (7) the primal lagrangian is:

$$L_p(\alpha,\beta,\chi,\gamma) = \frac{1}{2}\alpha^T Q\alpha - 1^T\alpha + \gamma y^T\alpha - \beta^T\alpha + \chi^T(\alpha - C) \tag{8}$$

where α has to be minimized. Thus, the stationary conditions are:

$$y^T\alpha = 0, \alpha \geq 0 \text{ and } C - \alpha \geq 0 \quad \text{(primal feasibility)} \tag{9}$$

$$Q\alpha - 1 + \gamma y - \beta + \chi = 0, \ \beta \geq 0 \text{ and } \chi \geq 0 \quad \text{(dual feasibility)} \tag{10}$$

$$\beta^T\alpha = 0 \text{ and } \chi^T(\alpha - C) = 0 \quad \text{(complementary conditions)} \tag{11}$$

which are linear functions of α, γ, β and χ. Thus, one can obtain the solution of the SVM-QP problem by finding a non negative solution for the l equations (9) which also satisfies the l equations (9-11). Also, the dual of the QP problem by:

$$\underset{\alpha}{\text{Minimize}} \quad L_D(\beta,\chi,\gamma)=\frac{1}{2}(1-\gamma y+\beta-\chi)^T Q^{-1}(1-\gamma y+\beta-\chi)-\chi^T C \tag{12}$$

$$\text{Under the constraints} \quad \beta \geq 0, \tag{13}$$

$$\chi \geq 0, \tag{14}$$

which has the same KKT conditions and solution.

4. Algorithms for quadratic optimization problems

There are two main types of algorithms for the resolution of QP problems:

- Interior point methods that aim for the complementary conditions by keeping primal and dual feasibility at the same time.
- Active set methods, which are divided in primal and dual. Primal methods aim for dual feasibility by keeping primal feasibility and complementary conditions and the dual methods work with primal feasibility by keeping dual feasibility and the complementary conditions. The dual methods have the condition that the Q matrix must be a positive definite matrix.

5. Large scale SVM-QP implementation

Optimization algorithms to solve the SVM-QP (6)-(7), described in the preceding section are operational to solve problems of less than 2000 examples. Beyond this limit, depending on memory and processing capacities, it is not possible to use any QP technique without some modifications. For large-scale data bases, it is difficult to calculate and store the matrix Q of the vector products $k(x_i,x_j)$, because of data processing limitations. Consequently, one must find more effective methods for this kind of problems and be able to find the optimal solution in the minimum time with a moderate request of data-processing resources.

5.1. Decomposition technique

Different authors, [1], [9], [10] and [11], introduced the idea to break up the SVM-QP problem, into smaller sub problems which are easier to deal with. All those strategies consider two key points:

- Optimality conditions that make possible to check if the algorithm has optimally solved the problem, for the SVM-QP problem and its optimality conditions.
- Strategy of implementation, that defines the implementation of the objective function associated with variables violating optimality conditions, if a particular solution is not the global solution.

The solution of the SVM-QP (6)-(7) is optimal if and only if KKT conditions (9-11) are satisfied knowing that matrix Q is semi-definite positive. KKT conditions have a simple form to check them, thus, the SVM-QP problem can be solved when for any $i = 1,...,l$:

$$\alpha_i = 0 \rightarrow y_i g(\mathbf{x}_i) > 1 \quad (15)$$

$$0 < \alpha_i < C \rightarrow y_i g(\mathbf{x}_i) = 1 \quad (16)$$

$$\alpha_i = C \rightarrow y_i g(\mathbf{x}_i) < 1 \quad (17)$$

with $g(x_i)$ the argument of the sign decision function (5):

$$g(\mathbf{x}_i) = \sum_{j=1}^{l} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_i) + b \quad (18)$$

In order to incorporate optimality conditions, implementation strategy must take into account the fact that a significant part of Lagrange multipliers of α_i are equal to zero in the solution. In a similar way to primal active set methods, a solution is to divide the training set in active set A, also known as working set, and its complement N. Then, it is possible to rewrite the SVM-QP problem as follows:

$$\underset{\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_N}{\text{Minimize}} \quad q(\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_N) = \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_N\end{bmatrix}^T \begin{bmatrix}\mathbf{Q}_{AA} & \mathbf{Q}_{AN} \\ \mathbf{Q}_{NA} & \mathbf{Q}_{NN}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_N\end{bmatrix} - \begin{bmatrix}\mathbf{1}_A \\ \mathbf{1}_N\end{bmatrix}^T \begin{bmatrix}\boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_N\end{bmatrix}$$

$$\text{Under the constraints} \quad \begin{bmatrix}\mathbf{y}_A \\ \mathbf{y}_N\end{bmatrix}^T \begin{bmatrix}\boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_N\end{bmatrix} = 0,$$

$$\begin{bmatrix}\mathbf{0}_A \\ \mathbf{0}_N\end{bmatrix} \le \begin{bmatrix}\boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_N\end{bmatrix} \le \begin{bmatrix}\mathbf{C}_A \\ \mathbf{C}_N\end{bmatrix},$$

in which we can replace any $i \in A$ by any $j \in N$, without modifying the cost function. The main idea is to have only the support vectors in the active set A.
In this manner, the inactive set N is formed by zero-multipliers α_N.

Algorithm 1. Decomposition algorithm of the SVM-QP.

1. Election of an active unit initial A of size n_A.
2. Solve the QP (6)-(7), defined by active set A.
3. While there exist any $j \in N$ violating $y_j g(\mathbf{x}_j) > 1$.
   a. shift the n_A most erroneous vectors $\mathbf{x}_j$ to active set A,
   b. shift all vectors $\mathbf{x}_i$ with $\alpha_i = 0$, $i \in A$, to inactive set N, and return to step 2

The solution of this problem will be the solution of the SVM-QP problem if it verifies the optimality conditions (15)-(17), in particular $j_j g(x-j) > 1$, $j \in N(\alpha_j = 0)$. If it is not the case, then α_j, which corresponds to x_j, must be different to zero and, consequently, it is necessary to shift it to active set A. Certainly, we can make the same

for vectors x_i, associated with $\alpha_i = 0$ to N. The algorithm (1) summarizes the decomposition algorithm of the SVM-QP problem. To introduce better than a random initial working set, we can use a Gaussian search and accelerate the learning process [12]. An extreme of the decomposition is the sequential minimal optimization, SMO, suggested by Plat, [11], but, it is not useful for our work since it finds a sub optimal solution.

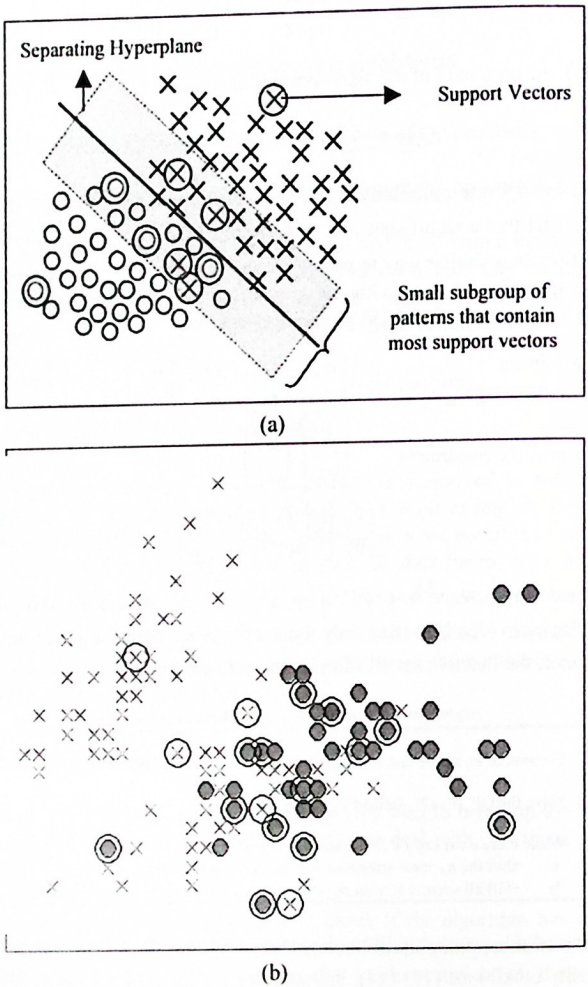

Fig. 1. Initialization strategies. (a) BCP initialization (b) Kernel Perceptron initialization.

6. Initialization strategies

Solving large scale applications require a decomposition method that breaks up the original problem into smaller QP sub problems. The main disadvantage with this approach is that patterns for each sub problem are selected randomly (causing substantial difference in the learning rate) for the decomposition method. In order to obtain a better initialization working set (in order to accelerate the learning process) algorithms like Kernel-Perceptron and BCP are used, instead of using a random one. To perform the task we take advantage of the characteristics of these algorithms: find an optimal separating hyperplane for a given dataset, easy to implement and minimal processing and memory requirements. The combination between these algorithms and the decomposition approach ensures that the initial sub problem is conformed by candidate support vectors that were extracted before training the SVM, and consequentially, the training time is reduced. The general idea is to obtain a hyperplane well-classifying the original dataset, by means of the proposed algorithms, then, by simple Euclidean distance between the hyperplane w and the different patterns x, we get the closest to w and form the active set A. Due to geometrical characteristics of the clusters, we can demonstrate that if the length of A is greater enough, then support vectors are included, and thus, we have the best working set A to train the SVM (Refer to Figure 1).

6.1. Barycentric Correction Procedure

BCP is an algorithm based on geometrical characteristics for training a threshold unit [2]. It is very efficient training linearly separable problems and it was proven that the algorithm rapidly converges towards a solution [3]. Also, it is a free parameters algorithm offering very fast solutions for any kind of problems. The algorithm defines a hyperplane $\mathbf{w}^T \cdot \mathbf{x} + \theta = 0$ dividing the input space for each class. Thus, we can define: $I_1 = \{1,\ldots,N_1\}$ and $I_0 = \{1,\ldots,N_0\}$ where N_1 represents the number of patterns of target 1 and N_0 the number of patterns of target -1. Also, let $b = (b_0, b_1)$ be the barycenter of points C_1 and C_0, weighted by the positive coefficients $\alpha = (\alpha_1,\ldots,\alpha_{N_1})$ and $\mu = (\mu_1,\ldots,\mu_{N_0})$ referred as weighting coefficients [3]:

$$b_1 = \frac{\sum_{i \in I_1} \alpha_i p_i}{\sum_{i \in I_1} \alpha_i}, \quad b_0 = \frac{\sum_{i \in I_0} \mu_i m_i}{\sum_{i \in I_0} \mu_i} \tag{19}$$

The weight vector w is defined as a vector difference $\mathbf{w} = b_1 - b_0$. At each iteration, barycenter moves towards misclassified patterns. Increasing the value of particular barycenter implies hyperplane moves on that direction. For computing the bias term θ, let's define $\vartheta : \Re^n \to \Re$ such that $\vartheta(p) = -\mathbf{w} \cdot p$. The bias term is calculated as follows: $\theta = \frac{\max \vartheta_1 + \min \vartheta_0}{2}$ Assuming the existence of $J_1 \subset I_1$ and $J_0 \subset I_0$

that refer to misclassified examples of target (+1, -1), barycenter modifications are calculated by:

$$\forall_i \in J_1 \qquad \alpha(new)_i = \alpha(old)_i + \beta_i \tag{20}$$
$$\forall_i \in J_0 \qquad \mu(new)_i = \mu(old)_i + \delta_i$$

Where β and δ are defined as follows:

$$\beta = \max\left\{\beta_{--}, \min\left[\beta_{-}, \frac{N_1}{N_0}\right]\right\}, \quad \delta = \max\left\{\delta_{--}, \min\left[\delta_{-}, \frac{N_0}{N_1}\right]\right\}$$

According to [3], β_{--} and δ_{--} can be set to 1 and β_{-}, δ_{-} set to 30.

6.2. Perceptron algorithm

The Perceptron algorithm [5] is a first approach method to deal with linearly separable problems. It is an incremental algorithm that starts with a weight vector $\mathbf{w} = \mathbf{0}$. At each iteration, small modifications to $\mathbf{w}$ are performed until a solution is reached. Convergence is ensured in a finite number of iterations for linearly separable problems. In this research, we make some modifications to the original algorithm to treat non-linearly separable problems.

Algorithm 2. Perceptron algorithm.

1. Initialize the weight vector $\mathbf{w}$, b and choose a learning step η
2. While there exist $i: i \in N$ such that $f(\mathbf{x}_i) \neq y_i$
3. Update w and b values according to $\Delta w = w_{old} + (\eta/2)(y_i \cdot \mathbf{x}_i)$ and $\Delta b = b_{old} + (\eta/2)(y_i)$

6.3. Kernel Perceptron extension

The kernel Perceptron algorithm [6] deals with non-linearly separable datasets. Basically the algorithm defines the dual function: $f(x) = \sum_{i=1}^{N} \gamma_i y_i \{\phi(\mathbf{x}^T_i) \cdot \phi(\mathbf{x})\} + b$ Where γ is the set of dual variables to be updated and the dot product $(\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}))$ is replaced for the kernel function $k(\mathbf{x}_i, \mathbf{x})$.

7. Experimental Results

In this section, we present the comparison tables of the two initialization strategies and the two adapted QP-SVM implementations: a dual active set algorithm (*QP1*) and an interior point algorithm (*QP2*). Also, a comparison of these methods with a second interior point method called pr-loqo (*QP3*), [4] is presented. For every initialization

strategy, we coupled a decomposition algorithm: *BCP +QP*, and *KernelPerceptron+QP*,. We used seven benchmark problems commonly referred in the literature: iris (iri), sonar (son), Pima Indian diabetes (pid), Tic Tac Toe (tic), Phonemes (pho), adult (adu) and Shuttle (shu) data sets [7]. Experiments were done with a RBF function with $\gamma = 0.5$ for the kernel parameters and the regularization parameter C was set to 1000. These parameters were selected by means of cross-validation. Results shown in Tables 1 and 2 are the standard mean of about 1000 executions. The best results were obtained using the *QP1* approach and the two initialization strategies (BCP and Kernel Perceptron). Although the good results obtained, an important problem is noted: The *BCP + QP* strategy has a good performance with these data sets but, if a benchmark with a high degree of non-linearity is tested, it is not guaranteed that the initial working set contains a significant number of support vectors. It is because the approach is approximating the solution by a linear function (in the original space), while the solution of the SVM is reached in a high dimensional space (in feature space). The *KernPerceptron+QP* strategy does not have this problem, because the Kernel Perceptron algorithm looks for the solution in the same high dimensional space that the SVM algorithm.

Table 1. Time performance of BCP initialization and QP solvers

	N	BCP + QP1	BCP + QP2	BCP + QP3
iri	150	0.0123 sec	0.391 sec	5.201 sec
son	208	0.16 sec	6.213 sec	16.231 sec
pid	768	5.143 sec	25.450 sec	42.960 sec
tic	958	3.091 sec	28.456 sec	41.975 sec
pho	1027	3.328 sec	26.727 sec	43.150 sec
adu	5000	24.028 sec	129.2773 sec	348.152 sec
shu	13633	35.431 sec	92.692 sec	202.428 sec

Table 2. Time performance of Kernel Perceptron initialization and QP Solvers

	N	Kperceptr + QP1	Kperceptr + QP2	Kperceptr + QP3
iri	150	0.0136 sec	0.3673 sec	6.010 sec
son	208	0.1231 sec	5.878 sec	14.231 sec
pid	768	2.296 sec	22.086 sec	38.069 sec
tic	958	1.635 sec	24.876 sec	40.352 sec
pho	1027	3.250 sec	28.297 sec	53.150 sec
adu	5000	13.35 sec	128.301 sec	355.278 sec
shu	13633	19.825 sec	102.192 sec	262.628 sec

8. Conclusions

Support Vector Machines is a promising methodology used in different research areas. Moreover, the optimization of the SVM is a delicate problem due to computational and memory requirements. This research is focused in combining different initialization strategies and three different SVM-QP solvers in order to select one that improves the training time of SVMs. The comparison of the different proposals shows that using kernel Perceptron and BCP as initialization strategies has a good performance, but the

SVM-QP solver that minimizes the training time was *QP1* which corresponds to the dual active set algorithm. Although, it was shown that the *BCP* initialization could have limitations when training data sets with a high non-linearity degree.

References

[1] Vapnik, V. Computational Learning Theory. John Wiley & Sons (1998)
[2] Poulard, H., Stève, D. Barycentric Correction Procedure: A fast method of learning threshold unit. World Congress on Neuronal Networks I (1995) 710-713
[3] Poulard, H., Stève, D. A convergence theorem for Barycentric Correction Procedure. Technical Report 95180 of LAAS-CNRS (1995)
[4] González-Mendoza, M. Quadratic Optimization fine tuning for the learning phase of SVM. ISSADS 2005 11 (2005).
[5] Rossemblat, F. The Perceptron: a probabilistic model for information storage and organization in the brain. Psychological review (1958) 386-408.
[6] Kowalczyk, A.: Maximal margin Perceptron. MT Press, Cambridge (1999).
[7] Keogh, E., Blake, C. and Merz, C.J. UCI repository of machine learning databases. http://kdd.ics.uci.edu (1998).
[8] Fletcher, R. Practical Methods of Optimization John Wiley and Sons; 2nd edition, May (2000).
[9] Joachims, T. Making large-scale support vector machine training practical. In Advances in Kernel Methods: Support Vector Machines. B. Schlkopf, C. Burges, A. Smola editors, MIT press, Cambridge, MA, pp. 169-184. (1998).
[10] Osuna, Edgar. Support Vector Machines: Training and Applications. Report of the Center of Biological and Computational Learning MIT. Paper No. 144 (1997).
[11] Platt, J. C. Fast Training of support vector machines using sequential minimal optimization. In B. Schlkopf, C. J. C. Burges and A. J. Smola, editors. Advances in Kernel Methods Chapter 12, pages 185-208. MIT press (1998).
[12] González-Mendoza, M, Titli, A., Hernández-Gress, N. Boosting Support Vector Machines in Density Estimation Problems. IEEE Information Processing and Management of Uncertainty, IPMU 2002. Annecy, France, (2002).