Incremental Support Vector Machine Learning: a Local Approach

Liva Ralaivola and Florence d'Alché–Buc

Laboratoire d'Informatique de Paris 6,
Université Pierre et Marie Curie,
8, rue du Capitaine Scott, F-75015 Paris, France
{liva.ralaivola, florence.dalche}@lip6.fr

Abstract. In this paper, we propose and study a new on-line algorithm for learning a SVM based on Radial Basis Function Kernel: Local Incremental Learning of SVM or LISVM. Our method exploits the "locality" of RBF kernels to update current machine by only considering a subset of support candidates in the neighbourhood of the input. The determination of this subset is conditioned by the computation of the variation of the error estimate. Implementation is based on the SMO one, introduced and developed by Platt [13]. We study the behaviour of the algorithm during learning when using different generalization error estimates. Experiments on three data sets (batch problems transformed into on-line ones) have been conducted and analyzed.

1 Introduction

The emergence of smart portable systems and the daily growth of databases on the Web has revived the old problem of incremental and on-line learning. Meanwhile, advances in statistical learning have placed Support Vector Machines (SVM) as one of the most powerful family of learners (see [6, 16]). Their specificity lies on three characteristics: SVM maximizes a soft margin criterion, the major parameters of SVM (support vectors) are taken from the training sample and non linear SVM are based on the use of kernels to deal with high dimensional feature space without directly working in it.

However, few works tackle the issue of incremental learning of SVM. One of the main reasons lies on the nature of the optimization problem posed by SVM learning. Although there exist some very recent works that propose ways to update SVM each time new data are available [4, 11, 15], they generally imply to re-learn the whole machine.

The work presented here starts from another motivation: since the principal parameters of SVM are the training points themselves, and as far as a local kernel such as a gaussian kernel is used, it is possible to focus learning only on a neighbourhood of the new data and update the weights of concerned training data. In this paper, we briefly present the key idea of SVM and then introduce incremental learning problem. State of the art is shortly presented and discussed. Then, we present the local incremental algorithm or LISVM and discuss the model selection method to determine the size of the neighbourhood to be used at each step. Numerical simulations on IDA benchmark datasets [14] are presented and analyzed.

2 Support Vector Machines

Given a training set $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$, support vector learning [6] tries to find a hyperplane with minimal norm that separates the data mapped into a *feature space* Ψ *via* a nonlinear map $\Phi : \mathbb{R}^n \rightarrow \Psi$, where n denotes the dimension of vectors $\mathbf{x}_i$. To construct such a hyperplane, one must solve the following quadratic problem [3]:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(\mathbf{x_i}, \mathbf{x_j}) \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{l} \alpha_i y_i = 0 \quad \text{and} \quad 0 \leq \boldsymbol{\alpha} \leq C \tag{2}$$

with kernel function k defined as $k(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$. Solution of this problem provides $\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \Phi(\mathbf{x}_i)$, a real value b with the help of optimality conditions and the decision rule: $f(\mathbf{x}) = sgn(\mathbf{w} \cdot \Phi(\mathbf{x}) + b)$.

The most efficient practical algorithms to achieve the resolution of the latter problem implement a strategy of iterative subsets selection [9, 12, 13] where only the points in those subsets may see their corresponding lagrange multiplier change. This process of optimizing the global quadratic problem only on a subset of the whole set of variables is a key point of our algorihtm since we use such an optimization scheme on subsets defined as neighbourhoods of new incoming data.

3 Incremental Learning and SVM

In incremental learning, the training dataset is not fully available at the beginning of the learning process as in batch learning. Data can arrive at any time and the hypothesis has to be updated if necessary to capture the class concept. In the following, we suppose that data are drawn from a fixed but unknown distribution. We view this task as a first step towards learning drifting concepts, e.g. learning to classify data that are drawn from a distribution that change over time.

More formally, the problem considered here may be stated as follows. Let H be a hypothesis family (such as the gaussain kernel-based SVM family). Let F be a fixed but unknown distribution over $(\mathbf{x}, y)$ pairs, $\mathbf{x} \in X$ and $y \in \{-1, 1\}$. We suppose that at time t, a training pattern is randomly sampled from F. Let us define S_t the current observed sample at time t. The goal of incremental on-line learning is to find and update an hypothesis $h^t \in H$ using available examples in order to minimize generalization error.

The Kernel-Adatron algorithm [8] is a very fast approach to approximate the solution of the support vector learning and can be seen as a component-wise optimization algorithm. It has been succesfully tested by their authors to dynamically adapt the kernel parameters of the machine, doing model selection in the learning stage. Nevertheless, the only drawback of this work is that it should not be straightforward to extend this work to deal with drifting concepts.

Another approach, proposed in [15], consists in learning new data by discarding all past examples except support vectors. The proposed framework thus relies on the

property that support vectors summarize well the data and has been tested against some standard learning machine datasets to evaluate some goodness criteria such as *stability*, *improvement* and *recoverability*.

Finally, a very recent work [4] proposes a way to incrementally solve the global optimization problem in order to find the exact solution. Its reversible aspect allows to do "decremental" unlearning and to efficiently compute leave-one-out estimations.

4 Local Incremental Learning of a Support Vector Machine

We first consider SVM as a voting machine that combines the outputs of experts, each of which is associated with a support vector in the input space. When using RBF kernel or any kernel that is based upon the notion of neighbourhood, the influence of a support vector concerns only a limited area with a high degree. Then, in the framework of on-line learning, when a new example is available it should not be necessary to re-consider all the current experts but only those which are concerned by the localization of the input. In some extent, the proposed algorithm is linked with work of Bottou and Vapnik [1] about local learning algorithm.

4.1 Algorithm

We sketch the updating procedure to build h^t from h^{t-1} when the incoming data $(\mathbf{x_t}, y_t)$ is to be learned, $\mathcal{S}_{t-1}$ being the set of instances learned so far, and $\mathcal{S}_t = \mathcal{S}_{t-1} \cup \{(\mathbf{x_t}, y_t)\}$:

1. *Initialize lagrangian multiplier α_t to zero*
2. *If $y_t f^{t-1}(\mathbf{x}_t) > 1$ (point is well classified) then terminate (take h^{t-1} as new hypothesis h^t)*
3. *Build a working subset of size 2 with x_t and its nearest example in input space*
4. *Learn a candidate hypothesis g by optimizing the quadratic problem on examples in the working subset*
5. *If the generalization error estimation of g is above a given threshold δ, increase the working subset by adding the next closest point to x_t not yet in the current subset and return to step 4*
6. *h^t is set to g*

We stop constructing growing neighbourhoods (see Fig. 1) around the new data as soon as the generalization error estimation falls under a given threshold δ. A compromise is thus performed between complexity in time and the value of the generalization error estimate, as we consider that this estimate is minimal when all data are re-learned. The key point of our algorithm thus lies on finding the size of the neighbourhood (e.g. the number K of neighbours to be considered) and thus on finding a well suited generalization error estimate, what we will focus on in the next section.

To increase computational speed, and implement our idea of locality, we only consider, as shown in Fig. 1, a small band around the decision surface in which points may be insteresting to re-consider. This band is defined upon a real ε: points of class y_t for which $y_t f(\mathbf{x}_t) \leq 1$ and points of class $-y_t$ for which $y_t f(\mathbf{x}_t) \leq 1 + \varepsilon$ are in the band.

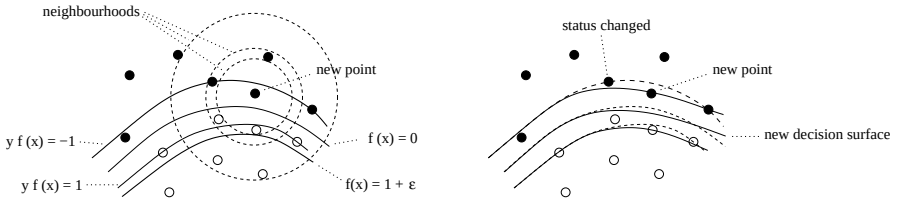

Fig. 1. *Left*: three neighbourhoods around the new data and the interesting small band of points around the decision surface parameterized by ε. *Right*: new decision surface and example of point whose status changed

4.2 Model Selection and the Neighbourhood Determinaton

The local algorithm requires to choose the value of K or the neighbourhood size. We have several choices to do that. A first simple solution is to fix K *a priori* before the beginning of the learning process. However the best K at time t is obviously not the best one at time $t + t'$. So it may be difficult to choose a single K suitable for all points. A second more interesting solution is therefore to determine it automatically through the minimization of a cost criterion. The idea is to apply some process of model selection upon the different hypothesis h_k^t that can be built.

The way we choose to select models consists in comparing them according to an estimate of their generalization error. One way to do that is to evaluate the estimation error on a test set and thus keeping K as the one for which E_{Test} is the least. In real problems, it is however not realistic to be provided with a test set during the incremental learning process. So this solution cannot be considered as a good answer to our problem.

Elsewhere, there exist some analytical expression of Leave-One-Out estimates of SVMs generalization error such as those recalled in [5]. However, in order to use these estimates, one has to ensure that the margin optimization problem has been solved exactly. The same holds for Joachims'$\xi\alpha-$estimators [10, 11]. This restriction prevents us from using these estimates as we only do a partial local optimization.

To circumvent the problem, we propose to use the bound on generalization provided by a result of Cristianini and Shawe-Taylor [7] for thresholded linear real-valued functions. While the bound it gives is large, it allows to "qualitatively" compare the behaviours of functions of the same family. The theorem states as follows:

Theorem 1. *Consider thresholding real-valued linear functions $\mathcal{L}$ with unit weight vectors on an inner product space X and fix $\gamma \in \mathbb{R}^+$. There is a constant c, such that for any probability distribution $\mathcal{D}$ on $X \times \{-1, 1\}$ with support in a ball of radius R around the origin, with probability $1 - \eta$ over ℓ random examples S, any hypothesis $f \in \mathcal{L}$ has error no more than*

$$\epsilon = err_{\mathcal{D}}(f) \leq B = \frac{c}{\ell}\left(\frac{R^2 + \|\boldsymbol{\xi}\|^2}{\gamma^2}\log^2 \ell + \log\frac{1}{\eta}\right) \tag{3}$$

where $\boldsymbol{\xi} = \boldsymbol{\xi}(\gamma, h, S) = (\xi_1, \xi_2, \ldots, \xi_\ell)$ is the margin slack vector with respect to f and γ defined as $\xi_i = \max(0, \gamma - y_i f(\mathbf{x}_i))$.

We notice that once the kernel parameter σ is fixed, this theorem, directly applied in the feature space Ψ_σ defined by the kernel k_σ, provides an estimate of generalization error for the machines we work on. This estimate is expressed in terms of a margin value γ, the norm of the slack margin vector $\boldsymbol{\xi}$ and the radius of the ball containing the data.

In order to use this theorem, we consider the feature space of dimension $d(K)$ defined by the Gaussian kernel with a fixed value for σ. In this space, we consider $\mathcal{L}$ with unit weight vectors. At step t, different functions h_k^t can be learnt with $k = 1, ..., K_{\max}$. For each k, we get a function f_k^t by normalizing the weight vector of h_k^t. f_k^t belongs to $\mathcal{L}$ and when thresholded provides the same outputs than h_k^t does. The theorem can then be applied to $f = f_k^t$ and data of S_t. It ensures that:

$$\epsilon \leq B(c(\gamma, \mathcal{L}), R_f, \xi_f, \gamma, t). \tag{4}$$

Hence, for each $k = 1...K_{\max}$, we can use this bound as a test error estimate.

However, as R_f is the radius of the ball containing the examples in the feature space [17], it only depends on the chosen kernel and not on k. On the contrary, ξ_f, defined as: $\xi_{i,k}^t = max(0, \gamma - y_i f_k^t(\mathbf{x}_i))$ is the unique quantity which differs among functions f_k. Slack vector ξ_k^t are thus sufficient to compare f_k functions, justifying our choice to use it as a model selection criterion. Looking at the bound, we can see that a value of γ must be chosen: in order to do that, we take a time-varying value defined as $\gamma_t = 1/\|\mathbf{w}_{t-1}\|$.

5 Experimental Results

Experiments were conducted on three different binary classification problems: *Banana* [14], *Ringnorm* [2] and *Diabetes*. Datasets are available at *www.first.gmd.de/~raetsch/*. For each problem, we tested LISVM for different values of the threshold δ. The main points we want to assess are the classification accuracy our algorithm is able to achieve, the appropriateness of the proposed criterion to select the "best" neighbourhood and the relevance of the local approach.

We simulated on-line incremental learning by providing the classifier with one example at a time, taken from a given training set. After each presentation, the current hypothesis is updated and evaluated on a validation set of size two thirds the size of the corresponding testing set. Only the "best test" algorithm uses the remaining third of this latter set to perform neighbourhood selection. An other incremental learning process called "rebatch" is also evaluated: it consists in realizing a classical SVM learning procedure over the whole dataset when a new instance is available. Experiments are run on 10 samples in order to compute means and standard deviations. Quantities of interest are plotted on Fig. 2 and Fig. 3. Table 1 reports the results at the end of the training process.

We led experiments for the same range of δ values in order to show that it is not difficult to fix a threshold that implies correct generalization performance. However, we must be aware that δ reflects our expectation of the error performed by the current hypothesis. Hence, smaller threshold δ should be preferred when the data are assumed to be easily separable (e.g. *Ringnorm*) while bigger values should be fixed when data are supposed to be harder to discriminate. This remark can be confirmed by the observation of *Banana* and *Diabetes* results. While the chosen values of δ lead to equivalent

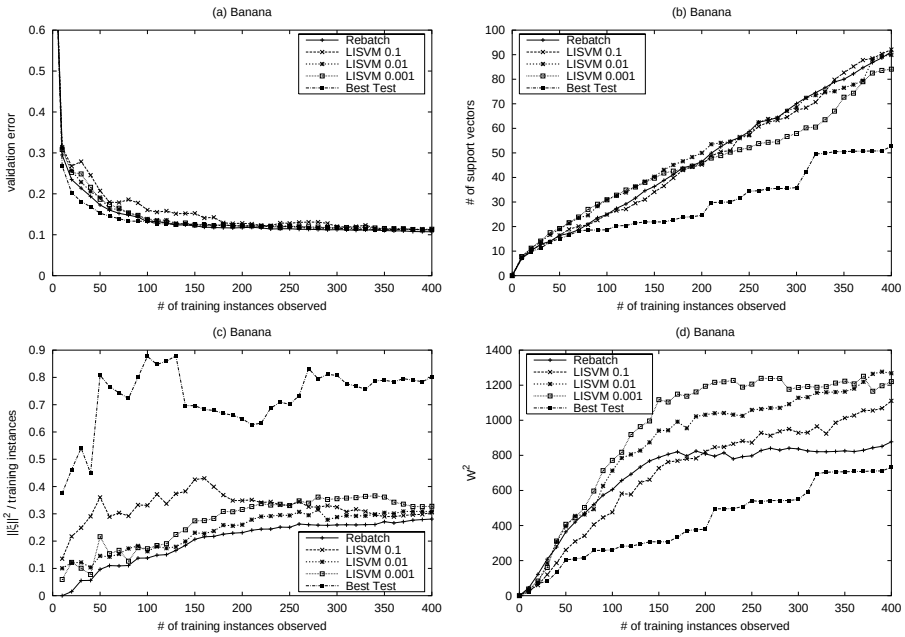

Fig. 2. Evolution of the machine parameters for the *Banana* problem during on-line learning with a potential band (see Fig. 1) defined by $\varepsilon = 0.5$

Table 1. Results on the three datasets. The potential band used (see Fig. 1) is defined by $\varepsilon = 0.5$. Classical SVM parameters are in the top-left cell of each table

Banana $C = 100$, $\sigma = 1$	validation error	Nb of Svs	$\mathbf{w}^2$	$\|\boldsymbol{\xi}(1,h,S)\|^2/\ell$
Batch/Rebatch	0.107 ± 0.004	91.2 ± 9.8	877 ± 166	0.281 ± 0.036
Best test	0.112 ± 0.013	52.7 ± 21.5	732 ± 232	0.801 ± 0.809
LISVM $\delta = 0.001$	0.113 ± 0.008	84.1 ± 18.8	1220 ± 291	0.328 ± 0.073
LISVM $\delta = 0.01$	0.113 ± 0.005	89.9 ± 9.4	1270 ± 244	0.309 ± 0.049
LISVM $\delta = 0.1$	0.111 ± 0.006	92.1 ± 7.9	1110 ± 220	0.305 ± 0.04

Ringnorm $C = 1e9$, $\sigma = 10$	validation error	Nb of Svs	$\mathbf{w}^2$	$\|\boldsymbol{\xi}(1,h,S)\|^2/\ell$
Batch/Rebatch	0.0263 ± 0.004	80.4 ± 8.5	3290 ± 701	$4.5e\text{-}6 \pm 4.5e\text{-}6$
LISVM $\delta = 0.001$	0.0272 ± 0.006	80.4 ± 8.5	3560 ± 809	0.007 ± 0.002
LISVM $\delta = 0.01$	0.0324 ± 0.008	65.9 ± 4	2180 ± 559	0.046 ± 0.012
LISVM $\delta = 0.1$	0.0628 ± 0.013	49.7 ± 5.8	1140 ± 500	0.155 ± 0.035

Diabetes $C = 10$, $\sigma = 20$	validation error	Nb of Svs	$\mathbf{w}^2$	$\|\boldsymbol{\xi}(1,h,S)\|^2/\ell$
Batch/Rebatch	0.228 ± 0.022	275 ± 7.1	382 ± 30.2	0.694 ± 0.017
LISVM $\delta = 0.001$	0.226 ± 0.023	197 ± 27.9	362 ± 70.2	0.646 ± 0.025
LISVM $\delta = 0.01$	0.229 ± 0.021	227 ± 36	382 ± 71.8	0.652 ± 0.031
LISVM $\delta = 0.1$	0.223 ± 0.023	226 ± 39.2	386 ± 66.4	0.652 ± 0.028

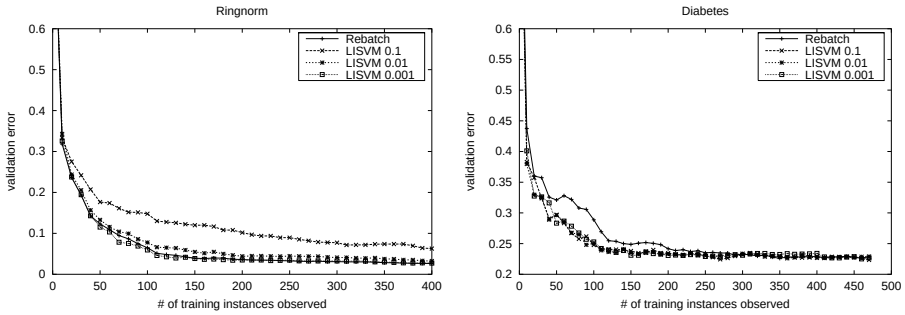

Fig. 3. Evolution of validation error for *Ringnorm* (*left*) and *Diabetes* (*right*)

(or bigger) complexity than the batch algorithm for equivalent validation error, other experiments with $\delta = 0.5$ show that LISVM obtains a lower complexity (55 SVs and $\|\mathbf{w}\|^2 = 677$ for *Banana*, 179 SVs and $\|\mathbf{w}\|^2 = 927$ for *Diabetes*) but with a degraded performance on the validation set (rates of 0.16 and 0.24 respectively). For *Ringnorm*, this observation can also be made in the chosen range of values $[0.001; 0.01; 0.1]$. Relaxing the value of δ leads to a lower complexity at the cost of a higher validation error. These experiments confirm that the relevant range of δ correponds to a balance between a low validation error and a small number of neighbours needed to reach the threshold at each step. CPU time measures provide means to directly evaluate δ for which the local approach sounds attractive. In the *Ringnorm* task for instance, CPU time is of 12.3 s for a large neighbourhood ($\delta = 0.001$) while it is reduced to 4.0 s and 1.9 s for respective smaller δ values of 0.01 and 0.1 and lower complexity.

Several curves reflecting the behaviour of the algorithm during time were drawn for the *Banana* problem. Same curves were measured for the other problems but are omitted for sake of space. The validation errors curves show the convergence of the algorithm. This behaviour is confirmed on the Fig. 2(c) where all the incremental algorithms exhibit a stabilizing value for $\|\boldsymbol{\xi}\|^2/\ell$. For LISVM and "rebatch" algorithm, the number of support vectors linearly increase with the number of new observed instances. This is not an issue, considering that the squared norm of the weight vector increases very much slower, suggesting that if the number of training instances had been bigger, a stabilization should have been observed. At last, let us consider the behaviour of the "best test" algorithm to LISVM on the *Banana* problem. This algorithm performs the SVM selection by choosing the size of the neighbourhood that minimizes the test error and thus is very demanding in terms of CPU time. Nevertheless, it is remarkable to notice that it reaches the same validation performance with twice less support vectors and a restricted norm of the weight vector, illustrating the relevance of the local approach.

6 Conclusion

In this paper, we propose a new incremental learning algorithm designed for RBF kernel-based SVM. It exploits the locality of RBF by re-learning only weights of training data that lie in the neighbourhood of the new data. Our scheme of model selection is

based on a criterion derived from a bound an error generalization from [7] and allows to determine a relevant neighbourhood size at each learning step. Experimental results on three data sets show very promising results and open the door to real applications. The reduction in terms of CPU time provided by the local approach should be especially important in case of availability of numerous training instances.

Next further works concern tests on large scale incremental learning tasks like text categorization. The possibility of the δ parameter to be adaptive will also be studied. Moreover, LISVM will be extended to the context of drifting concepts by the use of a temporal window.

References

1. L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992.
2. L. Breiman. Bias, variance and arcing classifiers. Technical Report 460, University of California, Berkeley, CA, USA, 1996.
3. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955–974, 1998.
4. G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Adv. Neural Information Processing*, volume 13. MIT Press, 2001.
5. O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing kernel parameters for support vector machines. Technical report, AT&T Labs, March 2000.
6. C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.
7. Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*, chapter 4 Generalisation Theory, page 68. Cambridge University Press, 2000.
8. T. Friess, F. Cristianini, and N. Campbell. The kernel-adatron algorithm: a fast and simple learning procedure for support vector machines. In J. Shavlik, editor, *Machine Learning: Proc. of the* 15^{th} *Int. Conf.* Morgan Kaufmann Publishers, 1998.
9. T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169–184. MIT Press, Cambridge, MA, 1998.
10. T. Joachims. Estimating the generalization performance of a svm efficiently. In *Proc. of the* 17^{th} *Int. Conf. on Machine Learning*. Morgan Kaufmann, 2000.
11. R. Klinkenberg and J. Thorsten. Detecting concept drift with support vector machines. In *Proc. of the* 17^{th} *Int. Conf. on Machine Learning*. Morgan Kaufmann, 2000.
12. E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines. In *Proc. IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285, 1997.
13. J.C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report 98-14, Microsof Research, April 1998.
14. G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. Technical Report NC-TR-1998-021, Department of Computer Science, Royal Holloway, University of London, Egham, UK, 1998.
15. N. Syed, H. Liu, and K. Sung. Incremental learning with support vector machines. In *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 1999.
16. V. Vapnik. *The nature of statistical learning theory*. Springer, New York, 1995.
17. V. Vapnik. *Statistical learning theory*. John Wiley and Sons, inc., 1998.