A New Model Selection Method for SVM

G. Lebrun[1], O. Lezoray[1], C. Charrier[1], and H. Cardot[2]

[1] LUSAC EA 2607, groupe Vision et Analyse d'Image, IUT Dépt. SRC,
120 Rue de l'exode, Saint-Lô, F-50000, France
{gilles.lebrun, c.charrier, o.lezoray}@chbg.unicaen.fr
[2] Laboratoire Informatique (EA 2101), Université François-Rabelais de Tours,
64 Avenue Jean Portalis, Tours, F-37200, France
hubert.cardot@univ-tours.fr

Abstract. In this paper, a new learning method is proposed to build Support Vector Machines (SVMs) Binary Decision Functions (BDF) of reduced complexity and efficient generalization. The aim is to build a fast and efficient SVM classifier. A criterion is defined to evaluate the Decision Function Quality (DFQ) which blendes recognition rate and complexity of a BDF. Vector Quantization (VQ) is used to simplify the training set. A model selection based on the selection of the simplification level, of a feature subset and of SVM hyperparameters is performed to optimize the DFQ. Search space for selecting the best model being huge, Tabu Search (TS) is used to find a good sub-optimal model on tractable times. Experimental results show the efficiency of the method.

1 Introduction

Data mining is considered as one of the challenging research fields of the 21^{th} century. Extracting knowledge from raw data is a difficult problem which covers several disciplines: Artificial Intelligence, Machine Learning, Statistics, Data Bases. Machine learning methods aim at providing classification methods which induce efficient decision functions. Among all possible inducers, SVMs have particular high generalization abilities and became very popular these last years. However decision functions provided by SVMs have a complexity which increases with the training set size [1,2,3]. Therefore, time processing with SVMs on huge datasets is not directly tractable. In recent years, there has been a lot of interest to improve learning methods using SVMs. One way is to optimize the SVM algorithm [1,4] to solve the associated quadratic problem. Other approaches use a simplification step to reduce the training set size [2,3,5,6,7]. For learning methods using SVM, model selection is critical. Many studies have shown that SVM generalization efficiency depends on the choices of SVMs parameters [8,9,10]. Other studies [11] have shown that multiclass SVMs are efficient if an efficient model selection is performed for each involved binary SVM. Therefore, as regards these considerations, new approaches aim at merging simplification step and model selection [5,3]. Although the SVM algorithms are lesser sensitive to curse of dimensionality [12], dimension reduction techniques can improve the

E. Corchado et al. (Eds.): IDEAL 2006, LNCS 4224, pp. 99–107, 2006.

efficiency of SVMs [8,10,12]. Our approach aims at unifying feature selection, simplification of training set and hyperparameters tuning as a complete model selection in order to produce efficient and low complexities BDFs with SVMs. For this new model selection method, a criterion named DFQ has been defined which takes into account the recognition rate of the BDF but also the number of support vectors (SV) and the number of features selected. For the simplification of the training set, the LBG algorithm used in vector quantization field [13] has been retained because it can produce good prototypes representing the initial dataset. Moreover the simplification level is controled by a single integer parameter whose values are few and can range from extreme simplification with only one prototype by class to no simplification. However, the proposed learning method is sufficiently general to be extended to other simplification methods. To have a proper tuning of hyperparameters and an accurate selection of relevant features, an adapted TS method is proposed for SVM model selection since usual SVM model selection have local minima [14]. Moreover TS has proved its suitability for such model selection problems [15,16]. The section 2 gives overviews and definitions used by our model selection method. The section 3 describe this new method and section 4 gives experimental results with it.

2 Overviews and Definitions

Support Vector Machines (SVM): SVMs were developed by Vapnik according to structural risk minimization principle from statistical learning theory [17]. Given training data (x_i, y_i), $i = \{1, \ldots, m\}$, $x_i \in \mathbb{R}^n$, $y_i \in \{-1, +1\}$, SVM maps an input vector x into a high-dimensional feature space $\mathbf{H}$ through some mapping function $\phi : \mathbb{R}^n \rightarrow \mathbf{H}$, and constructs an optimal separating hyperplane in this space. The mapping $\phi(\cdot)$ is performed by a kernel function $K(\cdot, \cdot)$ which defines an inner product in $\mathbf{H}$. The optimal solution α^* of corresponding convex quadratic programming problem [17] specifies the coefficients for the optimal hyperplane $w^* = \sum_{i=1}^{m} \alpha_i^* y_i \phi(x_i)$. The SV subset (i.e., $\alpha_i^* > 0$) gives the BDF $h(x) = \text{sign}(f(x))$ with $f(x) = \sum_{i \in \text{SV}} \alpha_i^* y_i K(x_i, x) + b^*$ where the threshold b^* is computed via the SVs [17]. SVMs being binary classifiers, several binary SVMs classifiers are combined to define a multi-class SVM scheme [11].

Vector Quantization (VQ): VQ is a classification technique used in the compression field [13]. VQ maps a vector x to another vector x' that belongs to m' *prototypes* vectors which is named *codebook*. The *codebook* S' is built from a training set S_t of size m ($m >> m'$). The algorithm must produce a set S' of prototypes which minimizes the distorsion $d' = \frac{1}{m} \sum_{i=1}^{m} \min_{1 \leq j \leq m'} d(x_i, x_j)$ ($d(.,.)$ is a $\mathcal{L}_2$ norm). LBG is an iterative algorithm [13] which produces 2^k prototypes after k iterates.

Decision Function Quality (DFQ): We consider that the DFQ of a given model θ depends on the recognition rate R_R but also on the complexity C_P of the DF h_θ when processing time is critical. Let $q(h_\theta) = R_R(h_\theta) - C_P(h_\theta)$ be the DFQ. For SVMs the complexity of the DF depends on the number of both SVs

and selected features. The empirical model we propose to model the complexity of a SVM BDF is: $C_P(h_\theta) = c_{p_1} \log_2(n_{SV}) + c_{p_2} \log_2(\text{cost}(\beta))$. β is a boolean vector of size n representing selected features. Constants c_{p_1} and c_{p_2} fix the trade-off between classification rate improvement and complexity reduction. Let κ_i denote the cost for the extraction of the i^{th} feature, the value of $cost(\beta)$ linked to the subset of selected features is defined by: $cost(\beta) = \sum \beta_i \kappa_i$. When these costs are unknown, $\kappa_i = 1$ is used for all features. Strictly speaking, a doubly of the number of SVs (extraction cost) is accepted in our learning method if it is related to a recognition rate increase of at least c_{p_1} (repectively c_{p_2}).

Tabu Search (TS): TS is a metaheuristic for difficult optimization problems [15]. TS belongs to iterative neighbourhood search methods. The general step, at the it iteration, consists in searching from a current solution θ^{it} a next best solution θ^{it+1} in a neighborhood. This new solution may be less efficient than the previous one, however it avoids local minimum trapping problems. That is why, TS uses short memory to avoid moves which might led to recently visited solutions (*tabu* solutions). TS methods generally use intensification and diversification strategies (alternately). In a promising region of space, the intensification allows extensive search to optimize a solution. The diversification strategy enables large changes of the solution to find quickly another promising region. Although the basic idea of TS is straightforward, the choice of solution coding, objective function, neighborhood, *tabu* solutions definition, intensification and diversification strategies, all depend on the application problem.

3 New Model Selection Method

The idea of our method is to produce fast and efficient SVM BDF using few features and SVs. A SVM is therefore trained from a small dataset S'_t representative of the initial training set S_t in order to reduce the complexity of the BDF and consequently training time. The LBG algorithm has been used to perform the simplification (reduction) of the initial dataset. Algorithm in Table 1 gives the details of this simplification[1]. As the level of simplification k cannot be easily fixed in an arbitrary way, a significant concept in our method is to regard k as variable. The optimization of SVM DFQ thus requires for a given kernel function K the choice of: the simplification level k, the feature subset β, the regularization constant C and kernel parameter σ. The search of the values of those variables is called model selection. Let θ be a model and k_θ, β_θ, C_θ, σ_θ be respectively the values of all the variables to tune. The search for the exact θ^* which optimizes the DFQ not being tractable, we decided to use tabu search as metaheuristic. Let the model θ be a vector of n' integer values[2] with $(\theta_1, \ldots, \theta_{n'}) = (\beta_1, \ldots, \beta_n, k, C', \sigma')$. A move for TS method is

[1] To speed up model selection, at each new value of k, the simplification result is stored for future steps which might use the same simplification level.

[2] $C_\theta = 2^{C'/2}$ with $C' \in [-10, \ldots, 20]$ (inspired by the *grid search* method [4]).

Table 1. Algorithm synopsis

Simplification(S,k)	**SVM-DFQ**(θ,S_l)
$S' \Leftarrow \emptyset$ **FOR** $c \in \{-1,+1\}$ $\mid$ $T = \{x \mid (x,c) \in S\}$ $\mid$ **IF** $2^k < \lvert T \rvert$ **THEN** $T' \Leftarrow \text{LBG}(T,k)$ $\mid$ **ELSE** $T' \Leftarrow T$ $\mid$ $S' \Leftarrow S' \cup \{(x,c) \mid x \in T'\}$ **RETURN** S'	$(S_t, S_v) \Leftarrow \text{Split}(S_l)$ $S'_t \Leftarrow \text{Simplification}(S_t, k_\theta)$ $h_\theta \Leftarrow \text{TrainingSVM}(S'_t, K_{\beta_\theta}, C_\theta, \sigma_\theta)$ $R_R \Leftarrow \frac{m_{-1}^{\texttt{correct}}}{2m_{-1}} + \frac{m_{+1}^{\texttt{correct}}}{2m_{+1}}$ $C_P \Leftarrow \text{Complexity}(h_\theta)$ $q(\theta) \Leftarrow R_R - C_P$
Intensification(θ^{it})	**Diversification**(θ^{it})

IF $q(\theta^{it}) > \eta_{\texttt{promising}} \cdot q(\theta_{\texttt{best-known}})$
THEN $\Theta_{\texttt{next}} \Leftarrow \text{ExtensiveSearch}(\theta^{it})$
ELSE $\Theta_{\texttt{next}} \Leftarrow \text{FastExtensiveSearch}(\theta^{it})$
$\theta^{it+1} \Leftarrow \text{BestNotTabu}(\Theta_{\texttt{next}})$
IF $q(\theta^{it+1}) > q(\theta_{\texttt{intensification}})$
THEN
$\theta_{\texttt{intensification}} \Leftarrow \theta^{it+1}$
$n_{\texttt{WithoutImprove}} \Leftarrow 0$
ELSE
$n_{\texttt{WithoutImprove}} \Leftarrow n_{\texttt{WithoutImprove}} + 1$
IF $n_{\texttt{WithoutImprove}} > n_{\max}$
THEN
$n_{\texttt{failure}} \Leftarrow n_{\texttt{failure}} + 1$
`stategy` $\Leftarrow$ `Diversification`
IF $n_{\texttt{failure}} > n_{\texttt{failure}}^{\max}$ **THEN** STOP
	$\delta \Leftarrow n_{\texttt{failure}+1}$
	$i \Leftarrow$ `SelectEligibleVariable`
	$\Theta_{\texttt{next}} \Leftarrow$ `TwoMove`$(\theta^{\texttt{it}}, \texttt{i}, \delta)$
	$\theta^{it+1} \Leftarrow \text{BestNotTabu}(\Theta_{\texttt{next}})$
	IF $q(\theta^{it+1}) > q(\theta_{\texttt{diversification}})$
	THEN
	$\theta_{\texttt{diversification}} \Leftarrow \theta^{it+1}$
	$n_{\texttt{diversification}} \Leftarrow n_{\texttt{diversification}} + 1$
	IF $n_{\texttt{diversification}} > n_{\max} \cdot n_{\texttt{failure}}$
	THEN
	$\theta^{it+1} \Leftarrow \theta_{\texttt{diversification}}$
	`stategy` $\Leftarrow$ `Intensification`

to add or substract δ ($\delta = 1$ for a basic move in intensification strategy) to one of those integer variables (i.e., $\theta_i^{it+1} = \theta_i^{it} \pm \delta$). The synopsis in Table 1 gives the details of the estimation of DFQ $q(\theta)$ from a model θ and a learning set S_l with $q(\theta) \equiv \text{SVM-DFQ}(\theta, S_l)$ the objective function to optimize. S_t, S_v sets produced by `Split` function ($|S_t| = \frac{2}{3}|S_l|$, $|S_v| = \frac{1}{3}|S_l|$) respectively indicate the bases used for SVM simplification step (training dataset) and for recognition rate estimation (validation dataset). This dissociation is essential to avoid the risk of overfitting when empirical estimation is used. For a given class $y \in \{+1,-1\}$, m_y represents the number of examples and $m_y^{\texttt{correct}}$ the correctly identified ones. This evaluation is more adapted when unbalanced class data are used. The kernel functions used is : $K_\beta(x_i, x_j) = \exp\left(-\sum_{l=1}^{n} \beta_l (x_{i,l} - x_{j,l})^2 / \sigma^2\right)$ with $x_{i,l}$ the l^{th} feature of example i. Feature selection is embedded in kernel functions by using β binary vectors ($\sigma = 2^{\sigma'/2}$ and σ' have the same range that C' in θ representation). The model selection TS algorithm has to deal with two kinds of problems. Firstly, testing all moves between two iterations with a great number of features can be time expensive. In particular, it is a waste of time to explore moves which are linked to features when the actual solution is not sufficiently promising. Therefore, intensification strategy focusing on moves

which are only linked to SVM hyperparameters or simplification level is more efficient to discover fastly new promising regions. Secondly, it is difficult for TS method to quickly escape from deep valleys of poor solutions when only using the short memory and resulting not taboo solutions. Using more diversified solutions can overcome this problem. This is dealt by increasing step size ($\delta > 1$) of moves and by forcing to use all types of moves (except feature selection moves for previous reason) in diversification strategy. Table 1 gives details of these two strategies. In intensification sysnopsis, `ExtensiveSearch` explores all eligible basic moves, whereas `FastExtensiveSearch` explores only eligible basic moves which are not related to feature selection (*i.e.* changing the value of β). $\eta_{\texttt{promising}}$ controls when the actual solution is considered as sufficiently promising. The set of all solutions θ which are *tabu* at the it iteration step of TS is: $\Theta^{it}_{tabu} = \{\theta \in \Omega \mid \exists\ i, t' : t' \in [1, \ldots, t],\ \theta_i \neq \theta_i^{it-1} \wedge \theta_i = \theta_i^{it-t'}\}$ with Ω the set of all solutions and t an adjustable parameter for the short memory used by TS (for experimental results $t = \sum_{i=1}^{n'} \max(\theta_i) - \min(\theta_i)$). `BestNotTabu` corresponds to the best solution on all possible moves from θ^{it} which are not *tabu* at this iteration. $n_{\texttt{max}}$ is the maximum number of intensification iterations for which no improve of the last best intensification solution ($\theta_{\texttt{intensification}}$) are considered as failure of the intensification strategy. In diversification sysnopsis, an eligible variable (those which do not have a relationship with features) is selected (`SelectEligibleVariable`) and a jump of $\pm\delta$ is performed by modifying the random selected variable in the actual solution. There are the two only explored moves (`TwoMove`) and this forces diversification. The jump size increases with the number of successive failures ($n_{\texttt{failure}}$) of the intensification strategy in order to explore more and more far regions. During the diversification iterations, the best visited solution is stored ($\theta_{\texttt{diversification}}$) and selected as the start solution for the next intensification step. At any time of TS exploration, if aspiration is involved, strategy automatically switch to intensification and the number of failures is reseted ($n_{\texttt{failure}} = 0$). The TS is stopped when the number of failures is higher than a fixed value and the best known solution is returned.

4 Experimental Results

We used the following datasets described in Table 2 (m, n_c and n are respectively the number of: examples, classes and features). *Adults* and *Shuttle* come from UCI repository [18], *Web* from [1] and *ClassPixels* from [7]. Learning and test sets contain respectively 2/3 and 1/3 of initial datasets. Test sets are used to estimate recognition rate (R_R) after model selection. For multiclass classification problem, the one-versus-all decomposition scheme is used. It produces n_c (number of class) binary classification problems [11]. For each one a model selection is realized. Figures 1(a) to 1(c) illustrate model selection experiments with Gaussian kernel for different

Table 2. Datasets description

bases	m	n_c	n
Shuttle	58000	6	9
Adults	45222	2	103
Web	49749	2	300
ClassPixels	224636	3	27

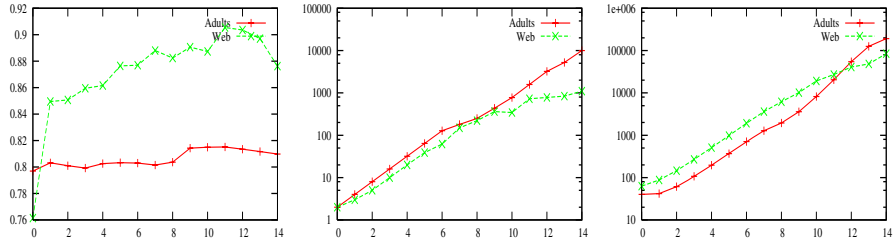


Fig. 1. Best recognition rate (a) and number of SVs (b) of BDF for a given simplification level k. *Grid search* method are used to select best model (C,σ). (c) gives total training time in seconds for grid search at each simplification level k.

simplification levels. Figure 1(a) shows that the level for which increasing training set size does not significantly improves recognition rate depends on dataset (*i.e.* importance of redundancy in dataset). Figure 1(c) shows that direct training with the whole dataset is time expensive for model selection. Moreover, complexity of decision function (fig. 1(b)) is directly linked to training set size of this one.

The objective of our learning method is to automatically select different parameters for producing SVM BDF which optimize the DFQ. However new parameters have been introduced and this can be problematic. Next experiments show how to deal with them. Our learning method is applied[3] on *Adults* dataset and table 3 (top) shows that an increase of $\eta_{\texttt{promising}}$ reduces the learning time without reducing quality of solution. Experiments with other datasets gave the same results and $\eta_{\texttt{promising}}$ can be fixed at 99%. Similar experiments have shown that a good compromise between learning time and quality of produced solution is $n_{\texttt{max}} = 5$ and maximum of accepted failure $n^{\texttt{max}}_{\texttt{failure}} = 5$.

Tables 3 (middle and bottom) illustrates the complexity evolution obtained with our learning method by using different penalties (c_{p_1} and c_{p_2}). Results show that higher penalties significantly reduce the number of SVs, the number of selected features and learning time while the recognition rate decrease is low if penalty is not too high. Of course good compromise depends on the considered application. Another interesting observation for a multiclass SVM scheme is that selected simplification levels could be different for each binary SVMs (*Shuttle* set in tab. 3). If training time are compared to the classical *grid search* methods without simplification of training set (fig. 1(c)), training time is greatly reduced (except for very low penality and feature selection) whereas our method preforms in addition feature and simplification level selection. Let n_k be the number of solutions θ examined by TS for which simplification level is equal to k. Global SVM training time of our method is $O(\sum n_k(2^k)^\gamma)$ with $\gamma \approx 2$. The examination of our method shows that n_k decreases while k increases. This effect increases when c_p values increases and explains the efficient training time of our method. In

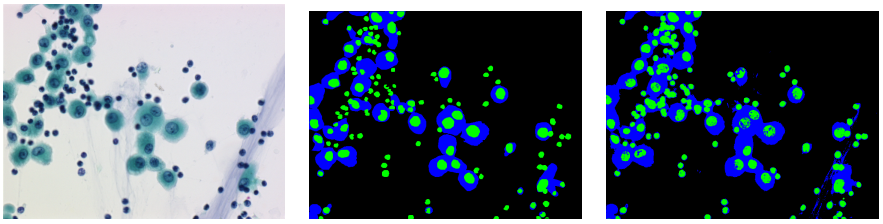
[3] Starting solution is: $k = \lfloor log_2(m/n_c)/3 \rfloor$, $C' = 0$, $\sigma' = 0$ and $n_{\texttt{feature}} = n$.

Table 3. Top: Influence of $\eta_{\texttt{promising}}$ for model selection (*Adults* dataset, $c_{p_1} = c_{p_2} = 10^{-3}$). Middle: Trade off between recognition rate and complexity (*Adult*(A) or *Web*(W) dataset, $\eta_{\texttt{promising}} = 99\%$). Bottom: Models selection for one-versus-all scheme (*Shuttle* dataset, $\eta_{\texttt{promising}} = 99\%$). For all tables $T_{\texttt{learning}}$ is the learning time for model selection in seconds, $n_{\texttt{feature}}$ is the number of selected features, DFQ is computed on the validation dataset (value of R_R in DFQ criterion) and selected model R_R is evaluated on a test dataset.

$\eta_{\texttt{promising}}$	$T_{\texttt{learning}}$	k	$log_{\sqrt{2}}(C)$	$log_{\sqrt{2}}(\sigma)$	n_{VS}	$n_{\texttt{feature}}$	R_R	DFQ
99.5 %	6654	6	0	0	50	9	79.3%	0.789
99.0 %	9508	1	-3	6	4	22	81.7%	0.803
98.0 %	160179	4	0	8	18	20	81.1%	0.804
95.0 %	195043	0	10	5	2	41	81.9%	0.803
0.0 %	310047	12	1	5	3286	44	81.8%	0.805

		with feature selection						without				
S	$c_{p_1} = c_{p_2}$	$T_{\texttt{learning}}$	k	n_{VS}	$n_{\texttt{feature}}$	R_R	DFQ	$T_{\texttt{learning}}$	k	n_{VS}	R_R	DFQ
A	0.0100	5634	0	2	44	81.5%	0.762	1400	0	2	79.4%	0.789
A	0.0020	16095	4	12	44	81.9%	0.793	2685	3	6	79.9%	0.800
A	0.0001	127096	10	764	55	81.8%	0.817	7079	13	5274	81.7%	0.811
W	0.1000	4762	1	3	44	82.2%	0.598	1736	1	3	84.1%	0.695
W	0.0100	25693	2	5	149	87.3%	0.846	4378	5	39	87.5%	0.835
W	0.0010	197229	9	506	227	89.7%	0.881	18127	11	730	90.4%	0.898

	c_{p_1}=c_{p_2}= **0.01**					c_{p_1}=c_{p_2}= **0**				
BDF	$T_{\texttt{learning}}$	k	n_{VS}	$n_{\texttt{feature}}$	R_R	$T_{\texttt{learning}}$	k	n_{VS}	$n_{\texttt{feature}}$	R_R
1-vs-all	207	4	7	2	99.85%	38106	15	127	3	99.83%
2-vs-all	67	0	2	1	99.93%	14062	10	20	3	99.95%
3-vs-all	45	0	2	1	99.94%	7948	11	38	3	99.95%
4-vs-all	152	5	9	2	99.91%	31027	14	63	4	99.94%
5-vs-all	44	3	2	1	99.98%	36637	7	13	2	99.96%
6-vs-all	113	2	5	1	99.97%	394	6	24	6	99.97%



(a) Microscopic image. (b) Expert segmentation. (c) Pixel classification.

Fig. 2. Pixel classification ($R_R = 90.1\%$) using feature extraction cost

our last experiments, pixel classification is performed for microscopical images. On such masses of data, the processing time is critical and we can assign a weight to each color feature of pixel: let $\kappa_i = n \cdot T_i/T$ be the weight with T_i the time

to extract the i^{th} color feature ($T = \sum_{i\in[1,\ldots,n]} T_i$). With our model selection method, pixel classification (fig. 2) can be performed with only 7 SVs and 4 color features (see [7] for further details).

5 Conclusions and Discussions

A new learning method is proposed to perform efficient model selection for SVMs. This learning method produces BDFs whose advantages are threefold: high generalization abilities, low complexities and selection of efficient features subsets. Moreover, feature selection can take into account feature extraction cost and many kinds of kernel functions with less or more hyperparameters can easily be used. Future works will deal with the influence of other simplification methods [2,6,5]. In particular, because QV methods can be time expensive with huge datasets.

References

1. Platt, J.: Fast training of SVMs using sequential minimal optimization, advances in kernel methods-support vector learning. MIT Press (1999) 185–208
2. Yu, H., Yang, J., Han, J.: Classifying large data sets using SVM with hierarchical clusters. In: SIGKDD. (2003) 306–315
3. Lebrun, G., Charrier, C., Cardot, H.: SVM training time reduction using vector quantization. In: ICPR. Volume 1. (2004) 160–163
4. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. Sofware Available at http://www.csie.ntu.edu.tw/~cjlin/libsvm (2001)
5. Ou, Y.Y., Chen, C.Y., Hwang, S.C., Oyang, Y.J.: Expediting model selection for SVMs based on data reduction. In: IEEE Proc. SMC. (2003) 786–791
6. Tsang, I.W., Kwok, J.T., Cheung, P.M.: Core vector machines: Fast SVM training on very large data sets. JMLR **6** (2005) 363–392
7. Lebrun, G., Charrier, C., Lezoray, O., Meurie, C., Cardot, H.: Fast pixel classification by SVM using vector quantization, tabu search and hybrid color space. In: CAIP. (2005) 685–692
8. Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing multiple parameters for support vector machines. Machine Learning **46** (2002) 131–159
9. Chapelle, O., Vapnik, V.: Model selection for support vector machines. In: Advances in Neural Information Processing Systems. Volume 12. (1999) 230–236
10. Fröhlich, H., Chapelle, O., Schölkopf, B.: Feature selection for support vector machines using genetic algorithms. IJAIT **13** (2004) 791–800
11. Rifkin, R., Klautau, A.: In defense of one-vs-all classification. JMLR **5** (2004) 101–141
12. Christianini, N.: Dimension reduction in text classification with support vector machines. JMLR **6** (2005) 37–53
13. Gersho, A., Gray, R.M.: Vector Quantization and Signal Compression. Kluwer Academic (1991)
14. Staelin, C.: Parameter selection for support vector machines. http://www.hpl.hp.com/techreports/2002/HPL-2002-354R1.html (2002)
15. Glover, F., Laguna, M.: Tabu search. Kluwer Academic Publishers (1997)

16. Korycinski, D., Crawford, M.M., Barnes, J.W.: Adaptive feature selection for hyperspectral data analysis. SPIE **5238** (2004) 213–225
17. Vapnik, V.N.: Statistical Learning Theory. Wiley edn. New York (1998)
18. Blake, C., Merz, C.: Uci repository of machine learning databases. advances in kernel methods, support vector learning. (1998)