Fast pixel classification by SVM using Vector Quantization, Tabu Search and Hybrid Color Space

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

[1] LUSAC EA 2607, groupe Vision et Analyse d'Image, IUT Dept. SRC, 120 Rue de l'exode, Saint-Lô, F-50000, France
{gilles.lebrun, c.charrier, o.lezoray, cyril.meurie}@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 (SVM) Binary Decision Function (BDF) of reduced complexity, efficient generalization and using an adapted hybrid color space. The aim is to build a fast and efficient SVM classifier of pixels. The Vector Quantization (VQ) is used in our learning method to simplify the training set. This simplification step maps pixels of the training set to representative prototypes. A criterion is defined to evaluate the Decision Function Quality (DFQ) which blends recognition rate and complexity of a BDF. A model selection based on the selection of the simplification level, of a hybrid color space and of SVM hyperparameters is performed to optimize this DFQ. Search space for selecting the best model being huge. Our learning method uses Tabu Search (TS) metaheuritics to find a good sub-optimal model on tractable times. Experimental results show the efficiency of the method.

1 Introduction

Pixels classification is commonly used as an initial step in color image segmentation schemes [1, 2] for the extraction of seeds. As for any classification problem, the choice of an inducer which produces efficient Decision Functions (DF) having good generalization performances is critical. Working with machine learning algorithm for pixel classification involves to take into account not only the recognition rate of the base inducer but also the processing time needed to perform a single pixel classification. In this paper, we are interested in SVMs and for those ones the processing time is only related to the complexity of the BDF. When a DF has an efficient recognition rate, but with a huge computing time per pixel, it cannot be directly used within the framework of pixel classification, expecially when processing time is critical. SVMs are powerful classifier having high generalization abilities [3]. However the BDF provided by SVM has a complexity which increases with the size of the training set [4, 5]. Therefore using SVMs on a huge pixel set is not directly tractable for pixel classification. To this aim we propose a new learning method which makes it possible to use SVMs within the pixel classification framework. This method uses the VQ principle [6] to simplify the training set and thus permits to reduce the complexity of the BDFs built by SVMs. the DFQ for the pixel classification depends on two terms: the DF recognition rate and the DF complexity. For pixel classification the DF complexity depends on the color space used and the number of Support Vectors (SV) (*cf.* section 5). The classical color space representation of a pixel is denoted by its RGB values, however, depending on the application, another more adapted color space can be chosen (XYZ,$L^*a^*b^*$,$L^*u^*v^*$,...). This choice is difficult and subjective, therefore it is more reliable to define a hybrid color space [1] which will be more adapted to the definition of a proper DF. For this reason, it is essential that our learning method selects a hybrid color space adapted to each BDF produced by the SVM. This hybrid color space is built by selecting a set of color components which can belong to any of the different classical color spaces [1]. The mechanism used in our method for the selection of the color components is similar to that usually used within the features selection framework [7]. For each BDF produced by SVM, our learning method must thus choose the values of the SVM hypermarameters, the simplification level of the training set and the hybrid color space in order to optimize the DFQ. Exhaustive search for model selection is not tractable, so we decided for this model selection to use TS metaheuristic because of it efficiency [8]. The combination of SVM, a simplification step by using VQ, a hybrid color space, a new criterion for the DFQ and a TS metaheuristic enables us to define a new learning method which produces in tractable times a efficient BDF which have a reduced complexity.

2 Support Vector Machines

The SVMs were developed by Vapnik and al. They are based on the structural risk minimization principle from statistical learning theory [3]. SVMs express predictions in terms of a linear combination of kernel functions centered on a subset of the training data, known as support vectors. Given the training data (x_i, y_i) , $i = \{1, \ldots, m\}$, $x_i \in \mathcal{R}^n$, $y_i \in \{-1, +1\}$, SVM maps the input vector x into a high-dimensional feature space $\mathbf{H}$ through some mapping functions $\phi : \mathcal{R}^n \to \mathbf{H}$, and builds 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 separating hyperplane given by a SVM is: $w \cdot \phi(x) + b = 0$. The optimal hyperplane is characterized by the maximal distance to the closest training data. The margin is inversely proportional to the norm of w. Thus computing this hyperplane is equivalent to minimize the following optimization problem: $\mathcal{V}(w, b, \xi) = \frac{1}{2}\|w\|^2 + C\left(\sum_{i=1}^{m} \xi_i\right)$ where the constraint $\forall_{i=1}^{m} : y_i \left[w \cdot \phi(x_i) + b\right] \geq 1 - \xi_i$, $\xi_i \geq 0$ requires that all training examples are correctly classified up to some slack ξ and C is a parameter allowing trading-off between training errors and model complexity. This optimization is a convex quadratic programming problem. Its whole dual [3] is to maximize the following optimization problem: $\mathcal{W}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$ subject to $\forall_{i=1}^{m} : 0 \leq \alpha_i \leq C \ , \ \sum_{i=1}^{m} y_i \alpha_i = 0$. The optimal solution α^* specifies the coefficients for the optimal hyperplane $w^* = \sum_{i=1}^{m} \alpha_i^* y_i \phi(x_i)$ and defines the subset SV of all SVs. An example x_i of the training set is a SV if $\alpha_i^* \geq 0$ in the optimal solution. The SVs subset gives the BDF h:

$$h(x) = \text{sign}(f(x)) \quad , \quad f(x) = \sum_{i \in SV} \alpha_i^* y_i K(x_i, x) + b^* \tag{1}$$

where the threshold b^* is computed via the unbounded SVs [3] (*i.e.* $0 < \alpha_i^* < C$). An efficient algorithm SMO [4] and many refinements [9, 10] were proposed to solve dual problem. SVM being binary classifiers, several binary SVM classifiers are induced for a multi-class problem. A final decision is taken from the outputs of all binary SVM [11].

3 Vector Quantization

The VQ is a classification technique used in the compression field [6]. 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_a of size m ($m >> m'$). The algorithm must produce a set S' of prototypes x' which minimizes the distorsion d' which is defined by: $d' = \frac{1}{m} \sum_{i=1}^{m} \min_{1 \leq j \leq m'} d(x_i, x_j)$ ($d(.,.)$ is a ℓ_2 norm). LBG is one of those algorithms [6] which can build this *codebook*. It is an iterative algorithm which produces 2^k prototypes after k iterates.

4 Hybrid Color Spaces

The pixels of a color image are digitized in (R, G, B) color space. However, this color space is not always the more appropriate for image processing problems and especially for pixel classification. There are many different color spaces and each one presents specific colorimetric, physical and physiological properties [1]. For our study, we have retained the most commonly used color spaces[3]: (X, Y, Z), (L^*, a^*, b^*), (L^*, u^*, v^*), (L_1, C, H_1), (Y_2, Ch_1, Ch_2), (I_1, I_2, I_3), (H_2, S, L_2), (Y_3, C_b, C_r). Moreover, in some experiments, it was shown that by combining color components from several color spaces, it is possible to build a hybrid color space more suitable than initial ones [1].

Let E be the space which regroups all n_E distinct color components from e different classical color spaces. By definition a hybrid color space H_E^β is composed of a set of n_β components from E and the vector β indicates which components from E are used (*i.e.* $i \in [1, \ldots, n_E]$, $\beta_i = 1$ if the i^{th} color component of the space E is used in H_E^β and $\beta_i = 0$ in the other case). For our study, $e = 9$, $n_E = 25$ and $E = (R, G, B, X, Y_1, Z, L^*, a^*, b^*, u^*, v^*, C, H_1, Y_2, Ch_1, Ch_2, I_1, I_2, I_3, H_2, S, L_2, Y_3, C_b, C_r)$. Then, the objective of our method is to find a hybrid color space H_E^β (the value of β) which improves the DFQ produced by SVM.

[3] We have added indices for some color components to differentiate them when being denoted by the same letter but not being identically computed.

5 Decision Function Quality

The DFQ q for 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. The DFQ q can be modelled by: $q(h_\theta) = R_R(h_\theta) - C_P(h_\theta)$. When the DF is built by SVM with a fixed kernel, the complexity of this DF depends on n_{SV} and β (H_E^β) . We chose to model $C_P(h_\theta)$ by: $C_P(h_\theta) = c_{p_1}\log(n_{SV}) + c_{p_2}\log(\text{cost}(\beta))$. Constants c_{p_1} and c_{p_2} are weighting coefficients which respectively represent the importance of the number of SVs and the choice of the hybrid color space $(\text{cost}(\beta))$ in the complexity of h_θ. The i^{th} color components $(i > 3)$ of a pixel are computed by linear or not linear transformation of the first three RGB components [1]. The time cost to compute a given color component is more or less expensive as regards the kind of transformation (linear or not, software or hardware). Let κ_i denote the transformation cost to compute the value of i^{th} color components, the value of $cost(\beta)$ linked to the hybrid color space H_E^β is defined by: $cost(\beta) = \sum_{i=1}^{n_E} \beta_i \kappa_i$.

6 Tabu Search

TS is a metaheuristic for difficult optimization problems. The roots of tabu search go back to the 1970s; it was first presented in its actual form by Glover [12]. 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 the neighbourhood. 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 creating cycles. The use of this short memory is helpful to avoid moves which might led to recently visited solutions (*tabu* solutions). Although the basic idea of TS is straightforward, the choice of solutions coding, objective function, neighbourhood, tabu solutions definition depends on the application problem.
Our problem is to choose an optimal model (solution) θ which can be represented by a set of integer variables $\theta = (\theta_1, \ldots, \theta_{n'})$ with $\forall i \in [1, \ldots, n'], \theta_i \in [min(\theta_i), \ldots, \max(\theta_i)]$ (*cf.* section 7). The objective function q to be optimized represents the quality of the BDF h_θ. One move in TS corresponds to adding $\Delta \in [-1, 1]$ to the value of θ_i, while preserving the constraints of the model which depends on it. From these constraints, the list of all possible neighboorhood solutions is computed. From these possible solutions the one which has the best DFQ and which is not *tabu* is chosen. The set of all Θ_{tabu}^{it} solutions θ which are *tabu* at the *it* iteration step of TS is defined as follow: $\Theta_{tabu}^{it} = \{\theta \in \Omega \mid \exists i, t' : t' \in \{1, \ldots, t\}, \theta_i = \theta_i^{it-t} \wedge \theta_i^{it-t} \neq \theta_i^{it-t+1}\}$ with Ω the set of all solutions and t an adjustable parameter for the short memory used by TS.

7 New learning method

When studying the SVM algorithm, one notices that processing time for SVM training quickly grows according to the size of the training base. For SMO algorithms, it is between $O(m^{1,6})$ and $O(m^{2,1})$ [4]. Moreover the number of SVs used by the BDF increases with the problem size. As the objective of our learning method is to produce a BDF of optimal qualitie (section 5), the increase in the number of SVs is only interesting if it is linked to a significant improvement of the recognition rate. The idea of our method is to train a SVM from a small data set representative of the initial one, 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 data set. Algorithm in Tab. 1 gives the details of this simplification. As the level of simplification k cannot be easily fixed

```
**Simplification**($S$,$k$)
$S' \Leftarrow \emptyset$
**FOR** $c = 1$ **TO** $n_c$
 | $T = \{x \mid (x,c) \in S\}$
 | **IF** $2^k < |T|$ **THEN** $T' \Leftarrow \mathrm{LBG}(T,k)$
 |                   **ELSE** $T' \Leftarrow T$
 | $S' \Leftarrow S' \cup \{(x,c) \mid x \in T'\}$
**RETURN** $S'$
```

```
**SVM-DFQ**($\theta$,$S_a$)
$(S_e, S_v) \Leftarrow \mathrm{Split}(S_a)$
$S'_e \Leftarrow \mathrm{Simplification}(S_e,k_\theta)$
$h_\theta \Leftarrow \mathrm{TrainingSVM}(S'_e,K_{\beta_\theta},C_\theta,\lambda_\theta)$
$R_R \Leftarrow 1-$ EmpiricalError($h_\theta$,$S_v$)
$C_P \Leftarrow$ Complexity($h_\theta$)
**DFQ** $\Leftarrow R_R - C_P$
```

Table 1. The Simplification algorithm (left) and the synopsis of SVM DFQ (right)

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 level of simplification k, the hybrid color space H_E^β, the constant of regularization C and the kernel parameters λ of K. The search of the values of these variables is called model search. Let θ be a model and k_θ, β_θ, C_θ, λ_θ be respectively the values of previous variables obtained from the model θ. The research of the exact value θ^* which optimizes the DFQ not being tractable, we decided to use tabu search as metaheuristic. To have a model θ easily usable by the TS, it must correspond to a vector of n' integer values. We have used the following equivalence: $(\theta_1, \dots, \theta_{n'}) = (\beta_1, \dots, \beta_{n_E}, k, C', \lambda'_1, \dots, \lambda'_{|\lambda|})$ with $C_\theta = 2^{C'}$, $C' \in [-5, \dots, 15]$ [9]. From this model θ, the function q which must be optimized by TS is $= q(h_\theta)$. The synopsis in Tab. 1 gives the details of the estimation of DFQ from a model θ and a training set S_a with $= q(h_\theta) = \text{SVM-DFQ}(\theta, S_a)$. S_e, S_v which is produced by Split function ($|S_e| = \frac{2}{3}|S_a|$, $|S_v| = \frac{1}{3}|S_a|$) respectively indicates the base used for training SVM and the estimate of the recognition rate. This dissociation is essential to avoid the risk of overfitting when the empirical error is used for the estimate of R_R. The SVM training step is made by using a SMO algorithm version present in the library Torch [10]. The kernel functions K_β used for training SVM are defined from a distance d_β: $d_\beta(x_i, x_j) = \sqrt{\sum_{l=1}^{n_E} \beta_l (x_i^l - x_j^l)^2}$.

It is identical to use ℓ_2 norm in the hybrid color space H_E^β for the design of BDF. For this study, only kernel: $K_\beta^L = {d_\beta}^2$ and $K_\beta^G = exp(-{d_\beta}^2/{\lambda_1}^2)$ are used (in TS model: $\lambda'_1 \in [-10, \dots, 10]$ and $\lambda_{1\theta} = 2^{\lambda'_1}$ [9]).

8 Experiments

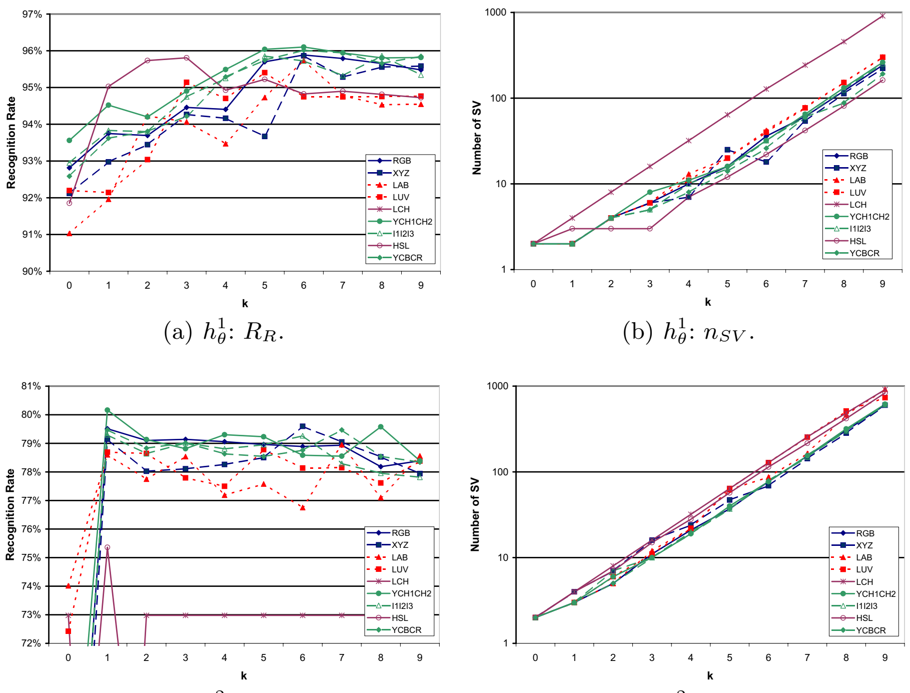
We applied our learning method for pixel classification of microscopic images of bronchial tumors [2]. The training and testing set S_a and S_t are built from four ground-thruth microscopic color images (RGB, 574*752 pixels). For each image, a manual segmentation is made by an expert: background (class 1), cytoplasm (class 2), nucleus (class 3). As the number of pixels in each class is not balanced in the images (1: 89%, 2: 7%, 3: 4%), only a subset of the pixels of classes 1 and 2 was selected by random to build S_a and S_t, so that each class has the same number of examples (≈ 60000 by class). Three training sets S_a^i (testing sets S_t^i) are built from the S_a (S_t) in order to produce binary decision problems (method *one against all* [11]). For each binary problem a model θ^i and BDF h_θ^i is built with our learning method. To avoid any biais for model selection the recognition rate of a BDF h_θ^i is evaluated from testing set S_t^i.
Figures 1(a) and 1(d) illustrate for each BDF h_θ^i with a kernel K_β^L (optimal value of C is searched) the evolution of the recognition rate and of the number of SVs according to the level of simplification k. That is done for all the classical color spaces retained. One can notice that improvement of R_R is obtained only for small values of k. Moreover, the choice of a color space which optimizes the DFQ depends to the trade-off between complexity and recognition rate. These remarks corroborate the choices which were made in the definition of our learning method.
Tables 3 illustrate results obtained with our learning method by using a kernel K_β^L and K_β^G. Table 2 gives the values of constants for all the configurations used. The column κ_i represents two cases: the first one is a microship transformation ($\kappa_i = 1$) and the second one is a software transformation ($\kappa_i = T_i/T$ with T_i the time to compute the color component i and $T = \sum_{i\in[1,\dots,n_E]} T_i$).

configuration	$\mathbf{c_{p_1}}$	$\mathbf{c_{p_2}}$	κ_i
A	0.0001	0.01	1
B	0.01	0.01	1
C	0.03	0.03	1
D	0.01	0.01	Ti/T

Table 2: values of constants

In tables 3 HCS indicates hybrid color spaces used by the BDF, Δt the training time, and in column DF is mentioned after θ the configuration which is used. These results show that our learning method produces BDF with reduced complexity and efficient in generalization. The choice of a specific hybrid color space for each BDF generally improves the recognition rate. The improvement with the use of kernel K_β^L is very significant ($\approx 5\%$) with h_θ^2. For this problem, the choice to use a kernel K_β^G in comparison with K_β^L does not improve the recognition but increases the BDF complexity. Indeed, $h(x) = d_\beta(x^*, x)^2 + b^*$ with $x^* = \sum_{i\in SV} \alpha_i^* y_i x_i$ when kernel K_β^L is used, then the number of SVs does not penalize the BDF complexity. However, although it seems logical to choose zero or very low values for c_{p_1}, results (Tab. 3: left, configuration A and B) show a significant increase in time for the selection of a model without significant improvement of the recognition rate. Those results (Tab. 3: configuration B and D) also show that the uses of κ_i constants allow to select a hybrid color space according to its cost. In particular, in the case of software implementation the

R, G, B components are mostly used and those requiring a nonlinear transformation lesser used, but the recognition rate still is as efficient.
As actually the whole process of microscopic images segmentation is software performed, then we have used the BDFs produced with the configuration D and a kernel K_{β}^{L}.



(a) h_{θ}^{1}: R_R. (b) h_{θ}^{1}: n_{SV}.

(c) h_{θ}^{2}: R_R. (d) h_{θ}^{2}: n_{SV}.

Fig. 1. Recognition rate and number of SVs in function of simplification level

9 Conclusions

A new learning method is proposed to build SVM binary decision functions which are efficient for pixel classification. This learning method produces BDF whose advantages for pixel classification problems are threefold: high generalization ability, low complexities and definition of a adapted hybrid color space. Future works will have to test this method on other pixel classification problems. It will also have to check the influence of several combination schemes of BDF, especially when the number of classes is higher. Later, it will also have to quantify the influence of other simplification methods and to compare other metaheuristics for model selection.

DF	R_R	n_{SV}	HCS	k	Δt
$h^1_{\theta,A}$	96.44%	2	ZC_r	2	144
$h^2_{\theta,A}$	85.99%	48	$Bb^*L_2H_1CH_2SC_b$	5	26574
$h^3_{\theta,A}$	90.48%	63	$Y_1L^*a^*u^*$	6	29540
$h^1_{\theta,B}$	96.50%	2	C_r	2	140
$h^2_{\theta,B}$	84.98%	18	$L^*b^*u^*CI_2S$	4	2352
$h^3_{\theta,B}$	89.79%	2	Gu^*C_b	1	147
$h^1_{\theta,C}$	96.50%	2	C_r	2	140
$h^2_{\theta,C}$	82.82%	3	SC_r	1	179
$h^3_{\theta,C}$	89.73%	2	v^*	1	140
$h^1_{\theta,D}$	95.66%	2	R	2	151
$h^2_{\theta,D}$	83.53%	2	$RBb^*SL_2C_r$	2	397
$h^3_{\theta,D}$	90.09%	4	GB	2	197

DF	R_R	n_{SV}	HCS	k	Δt
$h^1_{\theta,B}$	96.08%	5	RH_1	3	865
$h^2_{\theta,B}$	85.90%	10	$RXa^*H_1Y_2Y_3$	3	1239
$h^3_{\theta,B}$	90.43%	4	BYu^*v^*	2	708
$h^1_{\theta,C}$	95.66%	2	R	1	482
$h^2_{\theta,C}$	85.17%	10	$RH_1Y_2Y_3$	3	1223
$h^3_{\theta,C}$	89.47%	2	b^*	0	434
$h^1_{\theta,D}$	95.66%	2	R	1	440
$h^2_{\theta,D}$	85.78%	10	$RY_1Ch_1Y_3C_r$	3	1174
$h^3_{\theta,D}$	90.45%	5	GB	2	409

Table 3. Results with a microscopic image pixels set by using a kernel K^L_β (left) or a kernel K^G_β (right)

References

1. Vandenbroucke, N., Macaire, L., Postaire, J.G.: Color image segmentation by pixel classification in an adapted hybrid color space: application to soccer image analysis. Comput. Vis. Image Underst. **90** (2003) 190–216
2. Meurie, C., Lebrun, G., Lezoray, O., Elmoataz, A.: A comparison of supervised pixels-based color image segmentation methods. application in cancerology. In: WSEAS Transactions on Computers. Volume 2. (2003) 739–744
3. Vapnik, V.N.: Statistical Learning Theory. Wiley edn. New York (1998)
4. Platt, J.: Fast training of support vector machines using sequential minimal optimization, advances in kernel methods-support vector learning, MIT Press (1999) 185–208
5. Lebrun, G., Charrier, C., Cardot, H.: SVM training time reduction using vector quantization. In: ICPR. Volume 1. (2004) 160–163
6. Gersho, A., Gray, R.M.: Vector Quantization and Signal Compression. Kluwer Academic (1991)
7. Kudo, M., Sklansky, J.: Comparison of algorithms that select features for pattern classifiers. In: Pattern Recognition. Volume 33. (2000) 25–41
8. Hao, J.K., Galinier, P., Habib, M.: Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. In: Revue d'Intelligence Artificielle. Volume 13. (1999) 283–324
9. 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)
10. Collobert, R., Bengio, S.: SVMTorch: Support vector machines for large-scale regression problems. In: Journal of Machine Learning Research. Volume 1. (2001) 143–160
11. Hsu, C.W., Lin, C.J.: A comparison of methods for multiclass support vector machines. In: IEEE Transactions in Neural Networks. Volume 13. (2002) 415–425
12. Glover, F., Laguna, M.: Tabu search. Kluwer Academic Publishers, Boston MA (1997)