Journal of Systems Engineering and Electronics, *Vol. 17*, *No. 1*, *2006*, *pp. 200～205*

New approach to training support vector machine*

Tang Faming, *Chen Mianyun* & *Wang Zhongdong*
Dept. of Control Science & Engineering, Huazhong Univ. of Science and Technology, Wuhan 430074, P. R. China
(Received December 19, 2004)

Abstract: Support vector machine has become an increasingly popular tool for machine learning tasks involving classification, regression or novelty detection. Training a support vector machine requires the solution of a very large quadratic programming problem. Traditional optimization methods cannot be directly applied due to memory restrictions. Up to now, several approaches exist for circumventing the above shortcomings and work well. Another learning algorithm, particle swarm optimization, for training SVM is introduted. The method is tested on UCI datasets.

Key words: support vector machine, quadratic programming problem, particle swarm optimization.

1. INTRODUCTION

The support vector machine (SVM)[1] has been successful as a high-performance classifier in several domains including pattern recognition, data mining, and bioinformatics. It has strong theoretical foundations and good generalization capability. A limitation of the SVM design algorithm, particularly for large data sets, is the need to solve a quadratic programming (QP) problem involving a dense $n \times n$ matrix, where n is the number of points in the data set. Since QP routines have high complexity, SVM design requires huge memory and computational time for large data applications. Several approaches exist for circumventing the above shortcomings. These include simpler optimization criterion for SVM design, e.g., the linear SVM and the kernel adatron, specialized QP algorithms like the cojugate gradient method, decomposition techniques which break down the large QP problem into a series of smaller QP sub-problems, the sequential minimal optimization (SMO) algorithm[2] and its various extensions, Nystrom approximations[3], and greedy Bayesian methods[4]. A simple method to solve the SVM QP problem has been described by Vapnik, which is known as "chunking"[5].

Most of the above approaches have been employed successfully to solve the learning of SVM. In this paper, we introduce another algorithm, particle swarm optimization (PSO), as a training method of the SVM.

The PSO technique has been developed by Eberhart and Kennedy[6] and it is a simple evolutionary algorithm which differs from other evolutionary computation techniques in that it is motivated from the simulation of social behavior. PSO exhibits good performance in finding solutions to static optimization problems.

2. OVERVIEW OF SVM

Consider the problem of separating the set of training vectors belonging to two separate classes, (x_1, y_1), $\cdots, (x_l, y_l)$, where $x_i \in R^n$ is a feature vector and $y_i \in \{-1, +1\}$ a class label, with a hyperplane of equation $(w \cdot x) + b = 0$. Of all the boundaries determined by w and b, the one that maximizes the margin (Fig. 1) would generalize well as opposed to other possible separating hyperplanes.

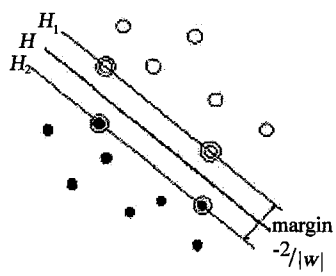

Fig. 1 The optimal separating hyperplane

A canonical hyperplane has the constraint for parameters w and b: $\min_{x_i} y_i((w \cdot x_i) + b) = 1$. A separating hyperplane in canonical form must satisfy the following constraints

$$y_i((w \cdot x_i) + b) \geqslant 1, i = 1, \cdots, l \qquad (1)$$

* The project was supported by the National Natural Science Foundation of China (79970025) and Advanced National Defence Found of China (00J15.3.3.JW0528).

The margin is $\frac{2}{\| w \|}$ according to its definition. Hence the hyperplane that optimally separates the data is the one that minimizes $\phi(w)=\frac{1}{2}\| w \|^2$.

The solution to the optimization problem can be obtained as follows: First, find the maximization solution to the following problem

$$\max_a W(\alpha) = \max_a\Big(\sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}a_i a_j y_i y_j (x_i \cdot x_j)\Big) \tag{2}$$

with constraints

$$\sum_{i=1}^{l}\alpha_i y_i = 0, \quad \alpha_i \geqslant 0, i = 1,\cdots,l \tag{3}$$

2.1 Nonlinear SVM

In the dual Lagrangian (2) we notice that the data points, x_i, only appear inside an inner product. To get a potentially better representation of the data we can map the data points into an alternative space, generally called feature space (a pre-Hilbert or inner product space) through a replacement

$$x_i \cdot x_j \rightarrow \phi(x_i)\cdot\phi(x_j)$$

The functional form of the mapping $\phi(x_i)$ does not need to be known since it is implicitly defined by the choice of kernel: $K(x_i, x_j)=\phi(x_i)\cdot\phi(x_j)$ or inner product in Hilbert space. With a suitable choice of kernel the data can become separable in feature space despite being non-separable in the original input space. Thus, whereas data for n-parity or the two spirals problem is non-separable by a hyperplane in input space it can be separated in the feature space defined by RBF kernels (giving an RBF-type network)

$$K(x_i, x_j) = e^{-\| x_i - x_j \|^2/2\sigma^2}$$

Many other choices for the kernel are possible, e.g.

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^d$$
$$K(x_i, x_j) = \tanh(\beta x_i \cdot x_j + b)$$

defining polynomial and feedforward neural network classifiers. Indeed, the class of mathematical objects which can be used as kernels is very general and includes, for example, scores produced by dynamic alignment algorithms. For binary classification with the given choice of kernel the learning task therefore involves maximization of the Lagrangian

$$\max_a W(\alpha) = \max_a\Big(\sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}a_i a_j y_i y_j K(x_i, x_j)\Big) \tag{4}$$

subject to constraints (3). After the optimal values of α_i have been found the decision function is based on the sign of

$$f(x) = \sum_{i=1}^{l}\alpha_i y_i K(x, x_i) + b \tag{5}$$

Since the bias, b, does not feature in the above dual formulation it is found from the primal constraints

$$b = -\frac{1}{2}\Big[\max_{\{i|y_i=-1\}}\Big(\sum_{x_j\in\{SV\}}^{m} y_j\alpha_j K(x_i, x_j)\Big) + \min_{\{i|y_i=+1\}}\Big(\sum_{x_j\in\{SV\}}^{m} y_j\alpha_j K(x_i, x_j)\Big)\Big] \tag{6}$$

When the maximal margin hyperplane is found in feature space, only those points which lie closest to the hyperplane have $\alpha_i > 0$ and these points are the support vectors. All other points have $\alpha_i = 0$. This means that the representation of hypothesis is solely given by those points which are closest to the hyperplane and they are the most informative patterns in the data.

2.2 Soft Margins and Allowing for Training Errors

An SVM can fit noise present in the training data leading to poor generalization. The effect of outliers and noise can be reduced by introducing a soft margin to remove the effect of outliers. Currently two schemes are possible. In the first (L_1 error norm) the learning task is the same as in (3,4) except for the introduction of the box constraint

$$0 \leqslant \alpha_i \leqslant C \tag{7}$$

while in the second (L_2 error norm) the learning task is the same as (3,4) except for addition of a small positive constant, λ, to the leading diagonal of the kernel matrix

$$K(x_i, x_j) \leftarrow K(x_i, x_j) + \lambda \tag{8}$$

C and λ control the trade-off between training error and generalization ability and are chosen by means of a validation set.

3. PREVIOUS METHODS FOR TRAINING SVM

All these tasks involve optimization of a quadratic Lagrangian and thus techniques from quadratic programming are most applicable including quasi-Newton, cojugate gradient and primal-dual interior point methods. Certain QP packages are readily applicable such as MINOS and LOQO. These methods can be used to train an SVM rapidly but they have the disadvantage that the kernel matrix is stored in memory. For small datasets this is practical and QP routines are the best choice, but for larger datasets alternative techniques have to be used. These split into two categories: techniques in which kernel components are evaluated and discarded during learning and working set methods in which an evolving subset of data is used. For the first category the most obvious approach is to sequentially update the α_i and this is the approach used by the kernel Adatron (KA) algorithm[7]. For binary classification (with no soft margin or bias) this is a simple gradient ascent procedure on (4) in which $\alpha_i \geqslant 0$ initially and the α_i are subsequently sequentially updated using

$$\alpha_i \leftarrow \beta_i \theta(\beta_i) \text{ where}$$

$$\beta_i = \alpha_i + \eta\left(1 - y_i \sum_{j=1}^{m} \alpha_j y_i K(x_i, x_j)\right) \quad (9)$$

and $\theta(\beta)$ is the Heaviside step function. The optimal learning rate η can be readily evaluated: $\eta = 1/K(x_i, x_i)$ and a sufficient condition for convergence is $0 < \eta K(x_i, x_i) < 2$. With the decision function (5) this method is very easy to implement and can give a quick impression of the performance of SVM on classification tasks. It is equivalent to Hildreth's method in optimization theory and can be generalized to the case of soft margins and inclusion of a bias[8]. However, it is not as fast as most QP routines, especially on small datasets.

3.1 Chunking and Decomposition

Rather than sequentially updating the α_i the alternative is to update the α_i in parallel but using only a subset or chunk of data at each stage. Thus a QP routine is used to optimize the Lagrangian on an initial arbitrary subset of data. The support vectors found are retained and all other data points (with $\alpha_i = 0$) discarded. A new working set of data is then derived from these support vectors and additional data points which maximally violate the storage constraints. This chunking process is then iterated until the margin is maximized. Of course, this procedure may still fail because the dataset is too large or the hypothesis modeling the data is not sparse (most of the α_i are non-zero, say). In this case decomposition[9] methods provide a better approach: these algorithms only use a fixed size subset of data with the α_i for the remainder kept fixed.

3.2 Decomposition and SMO

The limiting case of decomposition is the SMO algorithm of Platt in which only two α_i are optimized at each iteration. The smallest set of parameters which can be optimized with each iteration is plainly two if the constraint $\sum_{i=1}^{m} \alpha_i y_i = 0$ is to hold. Remarkably, if only two parameters are optimized and the rest kept fixed then it is possible to derive this analytical solution which can be executed using few numerical operations. The method therefore consists of a heuristic step for finding the best pair of parameters to optimize and use of an analytical expression to ensure the Lagrangian increases monotonically. For the hard margin case the latter is easy to derive from the maximization of δW with respect to the additive corrections a, b in $\alpha_i \rightarrow \alpha_i + a$ and $\alpha_j \rightarrow \alpha_j + b$ $(i \neq j)$. For the L_1 soft margin care must be taken to avoid violation of the constraints (7) leading to bounds on these corrections. The SMO algorithm has been refined to improve speed and generalized to cover the above three tasks of classification, regression and estimating densities. Due to its decomposition of the learning task and speed it is probably the method of choice for training SVM.

4. OVERVIEW OF PSO

In 1995, Kennedy and Eberhart first introduced PSO method. It is one of the optimization techniques and a

kind of evolutionary computation technique. The method has been found to be robust in solving problems featuring nonlinearity and nondifferentiability, multiple optima, and high dimensionality through adaptation, which is derived from the social-psychological theory. The features of the method are as follows.

(a) The method is developed from research on swarm such as fish schooling and bird flocking.

(b) It can be easily implemented, and has stable convergence characteristic with good computational efficiency.

Instead of using evolutionary operators to manipulate the particle (individual), like in other evolutionary computational algorithms, each particle in PSO flies in the search space with velocity which is dynamically adjusted according to its own flying experience and its companions' flying experience. Each particle is treated as a volumeless particle in D-dimensional search space.

Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (evaluating value) it has achieved so far. This value is called *pbest*. Another best value that is tracked by the global version of the particle swarm optimizer is the overall best value, and its location, obtained so far by any particle in the group, is called *gbest*.

The PSO concept consists of, at each time step, changing the velocity of each particle toward its *pbest* and *gbest* locations. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *gbest* locations.

For example, the ith particle is represented as $x_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,D})$ in the D-dimensional space. The best previous position of the i-th particle is recorded and represented as $pbest_i = (pbest_{i,1}, pbest_{i,2}, \cdots, pbest_{i,D})$. The index of best particle among all of the particles in the group is represented by the $gbest_d$. The rate of the position change (velocity) for particle i is represented as $v_i = (v_{i,1}, v_{i,2}, \cdots, v_{i,D})$. The modified velocity and position of each particle can be calculated using the current velocity and the distance from $pbest_{i,d}$ to $gbest_d$ as shown in the following formulas

$$v_{i,d}^{(t+1)} = w \cdot v_{i,d}^{(t)} + c_1 rand1()(pbest_{i,d} - x_{i,d}^{(t)}) + c_2 rand2()(gbest_d - x_{i,d}^{(t)}) \quad (10)$$

$$x_{i,d}^{(t+1)} = x_{i,d}^{(t)} + v_{i,d}^{(t+1)} \quad i = 1,2,\cdots,n \quad d = 1,2,\cdots,D \quad (11)$$

where n: number of particles in a group; D: number of members in a particle; t: pointer of iterations (generations); $v_{i,d}^{(t)}$: velocity of particle j at iteration t, $-V_d^{\max} \leqslant v_{i,d}^{(t)} \leqslant V_d^{\max}$; w: inertia weight factor; c_1, c_2: acceleration constant; $rand1()$, $rand2()$: random number between 0 and 1; $x_{i,d}^{(t)}$: current position of particle i at iteration t; $pbest_i$: *pbest* of particle i; *gbest*: *bgest* of the group.

In the above procedures, the parameter $V_d^{\max}$ determined the resolution, or fitness, with which regions be searched between the present position and the target position. If $V_d^{\max}$ is too high, particles might fly past good solutions. If $V_d^{\max}$ is too small, particles may not explore sufficiently beyond local solutions. In many experiences with PSO, $V_d^{\max}$ was often set at 10% ~ 20% of the dynamic range of the variable on each dimension.

The constants c_1 and c_2 represent the weighting of the stochastic acceleration terms that pull each particle toward *pbest* and *gbest* positions. Low values allow particles to roam far from the target regions before being tugged back. On the other hand, high values result in abrupt movement toward, or past, target regions. Hence, the acceleration constants c_1 and c_2 were often set to be 2.0 according to past experiences.

Suitable selection of inertia weight w in (10) provides a balance between global and local explorations, thus requiring less iteration on average to find a sufficiently optimal solution. As originally developed, w often decreases linearly from about 0.9 to 0.4 during a run. In general, the inertia weight w is set according to the following equation

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times t \quad (12)$$

where $iter_{\max}$ is the maximum number of iterations (generations), and t is the current number of iterations.

5. TRAINING SVM WITH PSO

Training of an SVM consists of determining the optimal value of non-negative multipliers α_i. In the pa-

per, we consider the training of a non-linear SVM, the training is expressed as a minimization of a dual quadratic form

$$\min_{a}\phi(\alpha)=\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}a_i a_j y_i y_j K(x_i,x_j)-\sum_{i=1}^{l}\alpha_i \quad (13)$$

subject to box constraints

$$0\leqslant\alpha_i\leqslant C,\ \forall\, i \quad (14)$$

and one linear equality constraint

$$\sum_{i=1}^{l}\alpha_i y_i=0 \quad (15)$$

Because the Lagrange multipliers α_i constitute a vector $\alpha=[\alpha_1,\alpha_2,\cdots,\alpha_l]$ in l-dimensional space, the optimization of QP problem (13) can be solved by PSO. Differing from the general PSO, all particles of the PSO training SVM must satisfy both constraints (14) and (15). Thus, the PSO algorithm must be improved.

According to box constraint (14), Eq. (10) is modified to

$$temp\_\, v_{i,d}^{(t+1)}=w\cdot v_{i,d}^{(t)}+c_1 rand1()(pbest_{i,d}-\alpha_{i,d}^{(t)})+c_2 rand2()(gbest_d-\alpha_{i,d}^{(t)})$$

$$v_{i,d}^{(t+1)}=\begin{cases}C-\alpha_{i,d}^{(t)}, & \alpha_{i,d}^{(t)}+temp\_\, v_{i,d}^{(t+1)}>C\\ -\alpha_{i,d}^{(t)}, & \alpha_{i,d}^{(t)}+temp\_\, v_{i,d}^{(t+1)}<0\\ temp\_\, v_{i,d}^{(t+1)}, & \text{othervise}\end{cases} \quad (16)$$

According to linear equality constraint (15)

$$\sum_{d=1}^{l}\alpha_{i,d}^{(t+1)}y_d=\sum_{d=1}^{l}\alpha_{i,d}^{(t)}y_d+\sum_{d=1}^{l}v_{i,d}^{(t+1)}y_d=\sum_{d=1}^{l}v_{i,d}^{(t+1)}y_d=\sum_{v_{i,d}^{(t+1)}y_d>0}v_{i,d}^{(t+1)}y_d-\sum_{v_{i,d}^{(t+1)}y_d<0}(-v_{i,d}^{(t+1)}y_d)=sumV_+-sumV_- \quad (17)$$

if (17) is not equal to zero, the velocity vector $v_i^{(t+1)}$ can be updated as follows

$$\text{if}\quad sumV_+>sumV_-,\text{then}\ v_{i,d}^{(t+1)}=\begin{cases}\dfrac{sumV_-}{sumV_+}v_{i,d}^{(t+1)} & \text{if } v_{i,d}^{(t+1)}y_d>0\\ v_{i,d}^{(t+1)} & \text{otherwise}\end{cases}$$

$$\text{else}\qquad v_{i,d}^{(t+1)}=\begin{cases}\dfrac{sumV_+}{sumV_-}v_{i,d}^{(t+1)} & \text{if } v_{i,d}^{(t+1)}y_d<0\\ v_{i,d}^{(t+1)} & \text{otherwise}\end{cases},\ d=1,\cdots,l \quad (18)$$

Thus, the Lagrange multipliers $\alpha_{i,d}^{(t+1)}$ satisfy both constraints (14) and (15).

The searching procedure of optimal or near optimal α are as shown below:

1. Initialize

(a) Set constants $p, C, c_1, c_2, iter_{\max}, v_0^{\max}, v_d, d, w_{\min}$, and $w_{\max}$.

(b) Set constants $t=0$. Set random number seed.

(c) Randomly initialize particle positions $\alpha_i^{(0)}\in R^l, 0\leqslant\alpha_{i,d}^{(0)}\leqslant C$, for $i=1,2,\cdots,p, d=1,2,\cdots,l$ where l is equal to the number of samples.

(d) If $\alpha_i^{(0)}$ violate the constraint (15), update $\alpha_i^{(0)}$ using (17), (18).

(e) Randomly initialize particle velocities $0\leqslant v_{i,d}^{(0)}\leqslant v_d^{\max}$, for $i=1,2,\cdots,p, d=1,2,\cdots,l$.

(f) Evaluate cost function values $\phi(\alpha_i^{(0)})$ using design space coordinates $\alpha_i^{(0)}$ for $i=1,2,\cdots,p$.

(g) Set $\phi_i^{pbest}=\phi(\alpha_i^{(0)})$ and $pbest_i=\alpha_i^{(0)}$ for $i=1,2,\cdots,P$.

(h) Set ϕ^{gbest} to minimal ϕ_i^{pbest} and $gbest$ to corresponding $\alpha_i^{(0)}$.

2. Optimize

(a) Update inertia weight w using (12)

(b) Update particle velocity vectors $v_i^{(t+1)}$ using (16)~(18).

(c) Update particle position vectors $\alpha_i^{(t+1)}$ using (11).

(d) Evaluate cost function values $\phi(\alpha_i^{(t+1)})$ using design space coordinates $\alpha_i^{(t+1)}$ for $i=1,2,\cdots,p$.

(e) If $\phi(\alpha_i^{(t+1)})<\phi_i^{pbest}$ then $\phi_i^{pbest}=\phi(\alpha_i^{(t+1)})$, $pbest_i=\alpha_i^{(t+1)}$ for $i=1,2,\cdots,p$.

(f) If $\phi(\alpha_i^{(t+1)}) < \phi^{gbest}$ then $\phi^{gbest} = \phi(\alpha_i^{(t+1)})$, $gbest = \alpha_i^{(t+1)}$ for $1,2,\cdots,p$.

(g) If all members of *gbest* fulfill the Karush-Kuhn-Tucker (KKT) optimality conditions of the QP problem, or the number of iterations, t, is up to $iter_{max}$, then go to 3. These KKT conditions are particularly simple

$$\alpha_i = 0 \Rightarrow y_i f(x_i) \geqslant 1, 0 < \alpha_i < C \Rightarrow y_i f(x_i) = 1$$
$$\alpha_i = C \Rightarrow y_i f(x_i) \leqslant 1 \quad (19)$$

(h) Increment t.

(i) Go to 2(a).

3. Report results

4. Terminate.

6. EXPERIMENTS

The PSO-SVM training algorithm is tested against the decomposition method on a series of benchmarks. PSO-SVM is written in C++, using Microsoft's Visual C++ 6.0 compiler, and does not make use of caching and shrinking implementation techniques. Chih-Jen Lin's LIBSVM[10] (version 2.6), a library for SVM and SVR, is used to test the decomposition method. The experiments were performed on three real-world datasets from the UCI[11]. The results of the experiments are shown in Table 1.

The first test set is the Cleveland heart dataset. The SVM is given 13 attributes chosen of the patients and asked to diagnose heart disease status. The chosen 13 attributes are all continuously valued. The second test is the wisconsin breast cancer dataset. Prediction task is to determine breast cancer is benign or malignant. Each sample is described by 9 integer-valued attributes. The last test is German credit dataset: classifying people described by 24 numerical attributes as good or bad credit risks.

Scaling samples before applying SVM is very important. The main advantage is to avoid attributes in greater numeric ranges dominate those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems. In our experiments, we scaled linearly each attribute to the range $[0,1]$. We trained two type of SVM using Gaussian RBF kernels

$$K(x_1,x_2) = \exp\left(-\frac{\|x_1 - x_2\|^2}{2}\right) \quad (20)$$

We chose a stopping tolerance of 0.005 to the stopping criterion (19) for quickly finding an optimal solution. The CPU time of all algorithms were measured on a 700 MHz Pentium Ⅲ processor running Windows 2000.

Table 1 The results of the experiments

Dataset $l \times q$	Algorithm	Training correctness/%	Time (CPU)/s	SVs
Cleveland heart 270×13	LIBSVM	100	1.2	270
	PSO	100	24.9	268
Wisconsin breast cancer 683×9	LIBSVM	100	2.1	97
	PSO	100%	34.2	87
German credit 1 000×24	LIBSVM	100	2.8	895
	PSO	100	41.3	892

The results of experiments show that training SVM with PSO is successful and accurate. Comparing to other training algorithms, the training speed of the PSO approach is slower than others. But, the PSO approach can consistently search a few support vectors less than others, and has better scaling ability.

7. CONCLUSION

In this paper we have shown that the standard particle swarm optimization can be used to train the SVM. The experimental results indicate that the PSO method is successful and accurate. But, comparing to previous algorithms, the main disadvantage of the PSO is its slow training speed. Thus, further work shall be done to optimize the training speed of the PSO using shrinking, caching and decomposition techniques.

(Continued to page 219)

REFERENCES

[1] Deng Yining, Manjunath B S. Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001, 23(8):800~810.

[2] Deng Yining, Kenney C, Moore M S, et al. Peer group filtering and perceptual color image quantization. *Proc. IEEE Int'l Symp Circuits and Systems*, 1999, 4:21~24.

[3] Udupa J K, Samarasekera Supun. Fuzzy connectedness and object definition: theory, algorithms, and applications in image segmentation. *Graphical Models and Image Processing*, 1996, 58(3): 246~261.

[4] Gevers T. Image segmentation and similarity of color-texture objects. *IEEE Trans. on Multimedia*, 2002, 4(4): 509~516.

[5] Udupa J K, Saha P K, Lotufo R A. Relative fuzzy connectedness and object definition: theory, algorithms, and applications in image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2002, 24(11): 1485~1500.

[6] Herman G T, Carvalho B M. Multiseeded segmentation using fuzzy connectedness. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001, 23 (5): 460~474.

[7] Mirmehdi M, Petrou M. Segmentation of color textures. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2000, 22 (2):142~159.

[8] Panjwani D K, Healey G. Markov random field models for unsupervised segmentation of textured color images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1995, 17(10): 939~954.

[9] Cheung WanFung, Chan YukHee. Color quantization of compressed video sequences. *IEEE Trans. on Circuits and Systems for Video Technology*, 2003, 13(3): 270~276.

[10] Karayiannis N B, Pai P I. Fuzzy vector quantization algorithms and their application in image compression. *IEEE Trans. on Image Processing*, 1995, 4(9): 1193~1201.

[11] Marques F, Kuo C-C J. Classified vector quantization using fuzzy theory. *IEEE International Conference on Fuzzy Systems*, 1992: 237~244.

[12] Dai Xiaoyan, Maeda J. Fuzzy based unsupervised segmentation of textured color images. *International Conference on Image Processing*, 2002, 3:24~28.

[13] Xie X L, Beni G. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991, 13(8): 841~847.

[14] http://maya.ece.ucsb.edu/JSEG

Zheng Yuanjie received Ph. D. degree from Institute of Image Processing & Pattern Recognition, Shanghai Jiaotong University, in 2006. Now he is a postdoctor research fellowship in the University of Pennsylvania, USA. His research interests are image segmentation and medical image analysis.

(Continued from Page 205)

REFERENCES

[1] Vapnik V. Statistical learning theory. *New York: Wiley*, 1998.

[2] Platt J C. Fast training of SVMs using sequential minimal optimization. In Schölkopf B, Burges C, Smola A (Eds.). Advances in kernel methods: support vector learning. *MIT Press*, 1998: 185~208.

[3] Williams C K I, Seeger M. Using the Nystrom method to speed up kernel machines. *Advances in Neural Information Processing Systems*, 2001, 13: 682~688.

[4] Tipping M E, Faul A. Fast marginal likelihood maximization for sparse bayesian models. *Proc. Int'l Workshop AI and Statistics*, 2003.

[5] Burges C J C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 1998, 2 (2): 1~47.

[6] Kennedy J, Eberhart R. Particle swarm optimization. *IEEE Int. Conf. Neural Networks*, 1995, *IV*: 1942~1948.

[7] Friess T T, Cristianini N, Campbell C. The kernel adatron algorithm: a fast and simple learning procedure for support vector machines. *15th Intl. Conf. Machine Learning*, *Morgan Kaufman Publishers*, 1998: 188~196.

[8] Luenberger D. Linear and nonlinear programming. *Addison-Wesley*, 1984.

[9] Osuna E, Girosi F. Reducing the run-time complexity in support vector machines. Schölkopf B, Burges C, Smola A (Eds.), Advances in Kernel Methods: Support Vector Learning, *MIT Press*, *Cambridge*, *MA*, 1999: 271~284.

[10] Chang Chih-Chung, Lin Chih-Jen. LIBSVM(V2.6): http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html,2003.

[11] Blake C L, Merz C J. UCI Repository of machine learning databases, http://www.ics.uci.edu/mlearn/MLRepository.html. Irvine, CA: University of California. 1998.

Tang Faming was born in 1977. He received the B. E. and M. E. degrees in control science & engineering form Huazhong University. of Science and Technology. Now he is studying for Ph. D. candidate in HUST. His main research interests include machine learning and data mining.

Chen Mianyun was born in 1937. He is a professor and a doctoral supervisor in Department of Control Science & Engineering of HUST. His main research interests include general systems, fuzzy control and grey system theory.

Wang Zhongdong was born in 1945. He is a professor in Department of Control Science & Engineering of HUST. His main research interest is control engineering.