Large Margin Kernel Pocket Algorithm

Jianhua Xu, Xuegong Zhang, and Yanda Li

Dept. of Autmation, Tsinghua University / State Key Lab of Intelligent Technology and Systems
Beijing 100084, China
xujianhua99@mails.tsinghua.edu.cn, zhangxg@tsinghua.edu.cn

Abstract

Two attractive advantages of SVM are the ideas of kernels and of large margin. As a linear learning machine, the original pocket algorithm can handle both linearly and nonlinearly separable problems. In order to improve its classification ability and control its generalization, we generalize the original pocket algorithm by using kernels and adding a margin criterion, and propose its kernel and large margin version, which can be referred to as large margin kernel pocket algorithm or LMKPA. The objective is to maximize both the number of correctly classified samples and the distance between the separating hyperplane and those correctly classified samples closest to the hyperplane, in the feature space realized with the kernels. This new algorithm only utilizes an iterative procedure to implement kernel idea and large margin simultaneously. For the linearly separable problems, LMKPA can find a solution that is not only without error, but also almost equivalent to that of SVM with the large-margin goal. For linearly non-separable problems, its performance is also very close to that of SVM. Experiments in numeral computation aspects show that the performance of LMKPA is close to that of SVM but the algorithm is much simpler.

I. INTRODUCTION

Although the original perceptron algorithm only can handle linearly separable problems [1-3], it is one of the simplest learning machines. To make perceptron suitable for non-separable cases, Gallant [4,5] proposed the pocket algorithm, which tries to minimize the number of samples being misclassified. He also proved the convergence theorem for integer or rational inputs (i.e., pocket convergence theorem). Muselli [6] further proved that the pocket convergence theorem still holds true for all possible inputs and that the pocket algorithm with ratchet finds an optimal solution within a finite number of iterations with probability one. In this paper, only the pocket algorithm with ratchet is studied and so we simply refer to it as pocket algorithm.

In recent years, support vector machine (SVM) proposed by Vapnik et al [7][8][9] is one of the most influential developments in the machine learning. Its two attractive notions are the idea of kernel and the idea of large margin. Now using kernel idea becomes an effective trick to design a nonlinear classifier. Smola et al. [10] pointed out that the large margin classifiers are robust with respect to samples and parameters.

By using the kernel idea, some authors extended the conventional linear methods and proposed their nonlinear forms with kernels, for example, the kernel Fisher discriminant analysis or KFD [11], the least square version for SVM or LS-SVM [12], and the kernel perceptron algorithm [13][14], etc.. Note that strictly speaking, the version of kernel perceptron algorithm from [13] is more similar to the potential function method (cf. [15]), while the version of kernel perceptron algorithm from [14] strictly complied with the derivation of classical perceptron algorithm. However the kernel perceptron algorithm can only deal with the linearly separable cases in the feature space too.

In order to realize the SVM ideas in a simpler way, some authors presented some iterative procedures, for instance, the kernel adatron [16], the voted perceptron algorithm with kernel [17], and the maximal margin perceptron [18]. But the first and second methods can only cope with linearly separable cases, while the last one is to solve the primary form of quadratic programming problem of SVM.

In this paper, to improve the classification ability of the original pocket algorithm and to control its generalization, we extend the linear pocket algorithm by using kernels and adding a margin criterion to define the large margin kernel pocket algorithm. Our objective is firstly to minimize the number of misclassified samples, and then maximize the minimal distance between the separating hyperplane and the correctly classified samples. For linearly separable cases in the feature space, we can obtain a solution with the largest margin and without misclassified samples, which can approach the performance of SVM. For other cases, its performance is also very close to that of SVM. In short, regardless of linearly or nonlinearly separable cases, our large margin kernel pocket algorithm utilizes a simple iterative procedure to fulfill the kernel idea and the large-margin idea simultaneously. Several experiment results demonstrate that the performance of large margin kernel pocket algorithm is very close to that of SVM.

II. BRIEF REVIEW OF THE PERCEPTRON AND POCKET ALGORITHM

Assume the training set

This work is supported by Natural Science Foundation of China, project No.69885004.

$$\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\} \tag{1}$$

is the set of samples from two different classes (ω_1, ω_2), where $\mathbf{x}_i \in R^n, y_i \in \{+1,-1\}$ and $y_i = +1$, if $\mathbf{x}_i \in \omega_1$; $y_i = -1$, if $\mathbf{x}_i \in \omega_2$, and l is the total number of samples in the training set.

For linear classifiers, the general form of discriminant function becomes

$$f(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b \tag{2}$$

where $\mathbf{w} \in R^n, b \in R$.

If it is assumed that the two classes are linearly separable, this hyperplane $f(\mathbf{x}) = 0$ can separate all samples correctly. Furthermore let $f(\mathbf{x}) > 0$ if $\mathbf{x} \in \omega_1$ and $f(\mathbf{x}) < 0$ if $\mathbf{x} \in \omega_2$.

For the original perceptron algorithm, the risk function is defined as

$$J_p(\mathbf{w}, b) = -\sum_{j \in \Gamma} ((\mathbf{w} \cdot \mathbf{x}_j) + b) y_j \tag{3}$$

where Γ is the set containing the subscripts of those samples that are misclassified by the separated hyperplane. If this risk function is minimized (to zero), it means that all training samples are classified correctly.

In order to minimize this risk function, an iterative procedure can be constructed based on gradient descent method, i.e.,

$$\begin{aligned} \mathbf{w}(t+1) &= \mathbf{w}(t) + \lambda_t \sum_{j \in \Gamma} \mathbf{x}_j y_j \\ b(t+1) &= b(t) + \lambda_t \sum_{j \in \Gamma} y_j \end{aligned} \tag{4}$$

where $\mathbf{w}(t)$ and $b(t)$ denote the weight vector and threshold at the tth iteration, and λ_t is the learning rate.

In the famous Rosenblatt perceptron algorithm [1][2], the learning rate is 1 and the weight vector and threshold are updated by single sample correction in each step, i.e.,

$$\text{if } y_k f(\mathbf{x}_k) \le 0, \text{ then } \mathbf{w} \Leftarrow \mathbf{w} + y_k \mathbf{x}_k, b \Leftarrow b + y_k \tag{5}$$

Moreover it was proven that this procedure converges to a solution within limited steps starting from any arbitrary initial values when the training samples are linearly separable (i.e., perceptron convergence theorem) (cf. [2][3][19]).

The classical perceptron algorithms can only cope with the linearly separable problems. For nonlinear problems, the solution in the iterative procedure oscillates. Many stop criteria are constructed to end the iteration. However the property of final solution is undetermined and behaves poorly in some cases.

In order to cope with the nonlinear cases, Gallant [4][5] proposed the pocket algorithm and proved its convergence theorem for integer or rational inputs (the so-called pocket convergence theorem). Its objective is to find a solution that can correctly classify a maximum number of the samples. The basic idea in the pocket algorithm is that it saves the weight vector and threshold with the longest consecutive run of correct classification trials in the perceptron algorithm. In the pocket algorithm with ratchet, if a new set of weight vector and threshold can classify more training samples than these previously saved, it saves such a new weight vector and threshold. This proper check effectively improves the performance of pocket algorithm. Muselli [6] further proved that the pocket convergence theorem holds for all possible input types and that the pocket algorithm with ratchet finds an optimal solution within a finite number of iterations with probability one. So pocket algorithm theoretically works well.

III. Brief Review of the Kernel Perceptron

The powerful classification ability of SVM and KFD inspired us to generalize the original perceptron algorithm by using the kernel idea [14]. Moreover the lager margin kernel pocket algorithm is based on the iterative procedure of kernel perceptron algorithm too.

Now suppose that training set (1) is not linearly separable, but can be separated by some nonlinear decision function. According to the ideas of SVM and KFD, we can map the samples into some new feature space F by some nonlinear transform:

$$\Phi : R^n \to F \tag{6}$$

and make the sample linearly separable in this space. Then in F, a linear discriminant function can be constructed, i.e.,

$$f^{\Phi}(\mathbf{x}) = f(\Phi(\mathbf{x})) = (\mathbf{w}^{\Phi} \cdot \Phi(\mathbf{x})) + \beta \tag{7}$$

where $\mathbf{w}^{\Phi}$ and β stand for the weight vector in the feature space respectively. From the theory of reproducing kernel it is concluded that any solution in feature space must lie in the span of all training samples in feature space [cf. 11]. Therefore the weight vector $\mathbf{w}^{\Phi}$ turns into the form of

$$\mathbf{w}^{\Phi} = \sum_{i=1}^{l} \alpha_i y_i \Phi(\mathbf{x}_i) \tag{8}$$

If we utilize the definition of kernel function satisfying the Mercer condition [8,9],

$$K(\mathbf{x}_i, \mathbf{x}) = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})) \tag{9}$$

the general form of discriminant functions with kernels becomes,

$$f^{\Phi}(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + \beta \tag{10}$$

where $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_l]^T$, $\alpha_i \in R^+$. Note that such kernel decision function is linear in the feature space and is nonlinear in the original attribute space.

Now assume that the kernel decision function (10) can separate all samples correctly, and that $f^{\Phi}(\mathbf{x}) > 0$ if

$\mathbf{x} \in \omega_1$ and $f^{\Phi}(\mathbf{x}) < 0$ if $\mathbf{x} \in \omega_2$. We can define an objective function with kernels as

$$J_p^{\Phi}(\boldsymbol{\alpha}, \beta) = -\sum_{i=1}^{l} \sum_{j \in \Gamma} \alpha_i y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_{j \in \Gamma} \beta y_j \ . \tag{11}$$

In order to minimize (11), we adopt a simply iterative procedure based on the gradient descent method for coefficients $\boldsymbol{\alpha}, \beta$, i.e.,

$$\begin{aligned} \alpha_i(t+1) &= \alpha_i(t) + \lambda_t \sum_{j \in \Gamma} K(\mathbf{x}_i, \mathbf{x}_j) y_i y_j \\ \beta(t+1) &= \beta(t) + \lambda_t \sum_{j \in \Gamma} y_j \end{aligned} \ . \tag{12}$$

Like Rosenblatt's perceptron algorithm, we still use $\lambda_t = 1$ and single sample correction, i.e.,

$$\begin{aligned} &\text{if} \quad y_k f^{\Phi}(\mathbf{x}_k) \le 0 \\ &\text{then} \quad \alpha_i \Leftarrow \alpha_i + y_i y_k K(\mathbf{x}_i, \mathbf{x}_k), \quad i = 1,2,...,l \\ &\qquad \text{and } \beta \Leftarrow \beta + y_k \end{aligned} \tag{13}$$

This is the kernel perceptron algorithm [14]. In [16], Friess et al illustrated a separating hyperplane obtained with kernel perceptron algorithm for the two spirals problem but did not give any detail. Guyon and Stork [17] designed a kernel perceptron algorithm that assigns $\alpha_k \Leftarrow \alpha_k + 1$ when the k th sample is misclassified, which strictly speaking is derived from the potential function method (cf. [15]). In our algorithm, we want to update all $\alpha_i, i = 1,..,l$ when the k th sample is misclassified. So our algorithm structure has some difference with the other theirs. Moreover the famous perceptron convergence theorem guarantees that our procedure converges to a solution within limited steps starting from any arbitrary initial values, when the training sample set is linearly separable in the feature space.

IV. Large Margin Kernel Pocket Algorithm

In this section, we propose our large margin kernel pocket algorithm and discuss its relation to SVM.

A. *Large Margin Kernel Pocket Algorithm*

Although the kernel perceptron algorithm possesses more powerful classification ability than the original one, it still can only deal with the linearly separable problems in the feature space. Moreover, from the viewpoint of statistical learning theory, in order to control the generalization of classifiers, we usually choose a kernel of less complexity. Thus in the feature space, it is still possible that the training samples are not completely separable.

As found by Glucksman [21], in many cases the separating hyperplane may lie very close to some training samples. Perceptron is such an example. Smola et al. [10] pointed out that the large margin classifiers are robust with respect to samples and parameters of classifiers.

In the original input space, the Euclidean distance from some sample point to the separated hyperplane is defined as $|f(\mathbf{x})| / \|\mathbf{w}\|_2$, where $\|\cdot\|_2$ is the 2-norm of a vector. Similarly in the feature space, the Euclidean distance becomes $|f^{\Phi}(\mathbf{x})| / \|\mathbf{w}^{\Phi}\|_2$, where $\|\cdot\|_2$ is the 2-norm of a vector in the feature space, i.e.

$$\|\mathbf{w}^{\Phi}\|_2 = \left(\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)\right)^{\frac{1}{2}} \tag{14}$$

Note that the kernel matrix satisfying Mercer condition is positive or semi-positive definite. Therefore 2-norm square of a vector is greater than or equal to zero.

Since it is supposed that $f^{\Phi}(\mathbf{x}_i) > 0$, $y_i = +1$ if $\mathbf{x} \in \omega_1$ and $f^{\Phi}(\mathbf{x}_i) < 0$, $y_i = -1$ if $\mathbf{x} \in \omega_2$, we can rewrite the definition of Euclidean distance from the ith sample point to the hyperplane as the form $\dfrac{y_i f^{\Phi}(\mathbf{x}_i)}{\|\mathbf{w}^{\Phi}\|_2}$. However, with this definition, for the some misclassified samples, the distances will be negative.

We define the margin as the minimal distance from the training samples to the hyperplane between two classes,

$$\rho = \min_i \frac{y_i f^{\Phi}(\mathbf{x}_i)}{\|\mathbf{w}^{\Phi}\|_2} \text{ if } y_i f^{\Phi}(\mathbf{x}_i) > 0 \ . \tag{15}$$

Now in order to cope with the nonlinear linear cases in the feature space we extend the linear pocket algorithm by using kernels, and in order to improve robustness of classifier we maximize the margin criterion (15). Thus a nonlinear algorithm with kernels and large margin is constructed. The procedure is shown Figure 1. We name this new algorithm as large margin kernel pocket algorithm, or LMKPA. Its objective is to minimize the number of misclassified samples while maximizing the minimal distance between the correctly classified samples and the separated hyperplane. It is important that we utilize a simply iterative procedure to implement these ideas.

The pocket convergence theorems [4-6] can guarantee that LMKPA finds the optimal solutions with probability one when the iteration increases. The computational complexity of LMKPA is almost identical with that of the linear pocket algorithms and kernel perceptron algorithm, since only minor extra computation is needed.

B. *Relationship with SVM*

In LMKPA, our objective is to find an optimal solution that can maximize both the number of correctly classified samples and the margin criterion. SVM is the most typical one in the family of large margin classifiers. Now we analyze the relation between LMKPA and SVM.

For linearly separable cases in the feature space, the SVM tries to find a solution that can classifies all training samples and makes the margin largest [7-9][22]. So the aim of SVM is same as that of LMKPA. Moreover maximizing the margin is identical to minimizing $\|\mathbf{w}\|_2^2$.

For the nonseparable cases in the feature space, SVM algorithm constructed the following pure quadratic programming problem [7-9][22],

$$\min \ \frac{1}{2}(\mathbf{w}^{\Phi} \cdot \mathbf{w}^{\Phi}) + C\sum_{i=1}^{l} \xi_i \tag{16}$$

s.t.

$$y_k((\mathbf{w}^{\Phi} \cdot \Phi(\mathbf{x}_k)) + \beta) \geq 1 - \xi_k \tag{17}$$
$$\xi_k \geq 0, \ k = 1,2,...l$$

where the second item in (16) is a penalty term for those samples violating $y_k((\mathbf{w}^{\Phi} \cdot \mathbf{x}_k) + b) \geq 1$.

In LMKPA, firstly we minimize the number of misclassified samples. This equals to say that the number of samples violating $y_k((\mathbf{w}^{\Phi} \cdot \mathbf{x}_k) + b) \geq 1$, i.e., the second term in (16), is minimized. Secondly we maximize the margin criterion. This causes that the first term of (16) is minimized and thus the margin between two classes is maximized. So in deed, LMKPA can be viewed as another realization of SVM, which is much simpler than SVM for it utilizes a simply iterative procedure.

V. EXPERIMENTS

In order to evaluate the performance of LMKPA and compare it with SVM, we design several experiments: a linear case, a linear case with some outliers and the two spirals problem.

A. *Linearly Separable Case*

As shown in Figure 2, there are about 80 two dimensional samples of two different classes, which are denoted by "+" and "o" respectively. The separating hyperplanes obtained by LMKPA and SVM are shown. The minimal Euclidean distances from the samples to hypreplanes for the two algorithms are 0.078 (LMKPA) and 0.072 (SVM).

Input: Training samples: $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\}$, number of iterations and $\varepsilon \in (0,1)$

Output: the pocket parameter vector and threshold: $\boldsymbol{\alpha}^*, \beta^*$

Temporary Variables:

$\boldsymbol{\alpha}, \beta$: weight vector and threshold in the iterative procedure

n: number of consecutive correct classification using $\boldsymbol{\alpha}, \beta$

n^*: number of consecutive correct classification using $\boldsymbol{\alpha}^*, \beta^*$

m: total number of training samples that $\boldsymbol{\alpha}, \beta$ correctly classify.

m^*: total number of training samples that $\boldsymbol{\alpha}^*, \beta^*$ correctly classify.

ρ: margin criterion using $\boldsymbol{\alpha}, \beta$

ρ^*: margin criterion using $\boldsymbol{\alpha}^*, \beta^*$

ρ_0: minimal margin or tolerance

Algorithm:

1. Set all temporary variables zeros
2. Randomly pick a sample: $(\mathbf{x}_k, y_k)$
3. If $\boldsymbol{\alpha}, \beta$ can correctly classify this sample, i.e., $f^{\Phi}(\mathbf{x}_k) > \rho_0$

   then

   3a: $n = n + 1$

   3b: If $n > n^*$, then

   3ba: Compute m by checking all samples and the margin criterion ρ

   3bb: If ($m > m^*$) OR ($m = m^*$ AND $\rho > \rho^*$), then

   3bba: Set $\boldsymbol{\alpha}^* = \boldsymbol{\alpha}, \beta^* = \beta$

   $n^* = n, m^* = m, \rho^* = \rho$

   3bbb: If all training samples are correctly classified (separable), then $\rho_0 = (1.0 + \varepsilon)\rho^*$.

   Otherwise

   3c: form a new $\boldsymbol{\alpha}, \beta$,

   $\alpha_i = \alpha_i + y_k K(\mathbf{x}_i, \mathbf{x}_k), \beta = \beta + y_k$

   3d: Set $n = 0$
4. End of this iteration. If the specified number of iterations has not been taken then go to 2.

Figure 1: Large Margin Kernel Pocket Algorithm (LMKPA).

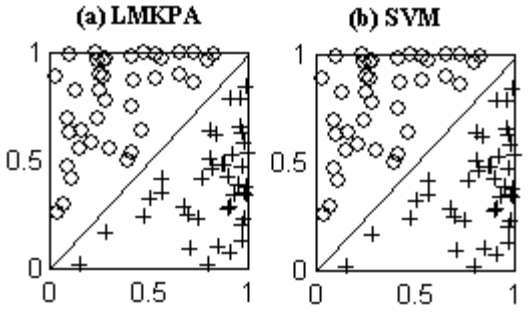

Figure 2. The separating hyperplanes of LMKPA (a) and SVM (b).

B. Linear Case with Two Outliers

When we randomly change the labels of two samples in the above example and take of them as two outliers, LMKPA finds a solution with two misclassified training samples, as shown in Figure 3. The margin is 0.076 in our algorithm, while 0.070 in SVM. The performance of LMKPA and SVM are still almost identical.

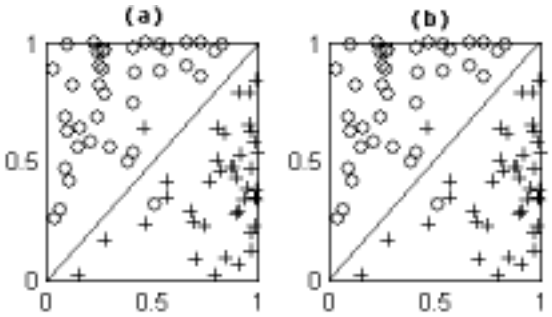

Figure 3. The decision plane of LMKPA (a) and SVM (b) when two outliers exist in the data set.

Actually if one takes the separation margin as the evaluation of the performance, our LMKPA even slightly overperforms SVM.

C. The Two Spirals Problem

For the two spirals problem, our task is to discriminate between two sets of sample points which lie on two spirals in a plane. In our example, each category includes 108 samples drawn by the symbols "+" and "." respectively in Figure 4. When using RBF kernel with $\sigma = 0.02$, kernel perceptron, LMKPA and SVM all classify the samples correctly. From Figure 4 we can see that the decision plane of kernel perceptron is not very smooth, while both LMKPA and SVM achieve the central and smooth separating hyperplanes. The difference between LMKPA and SVM on this problem is hardly distinguishable, if there are any.

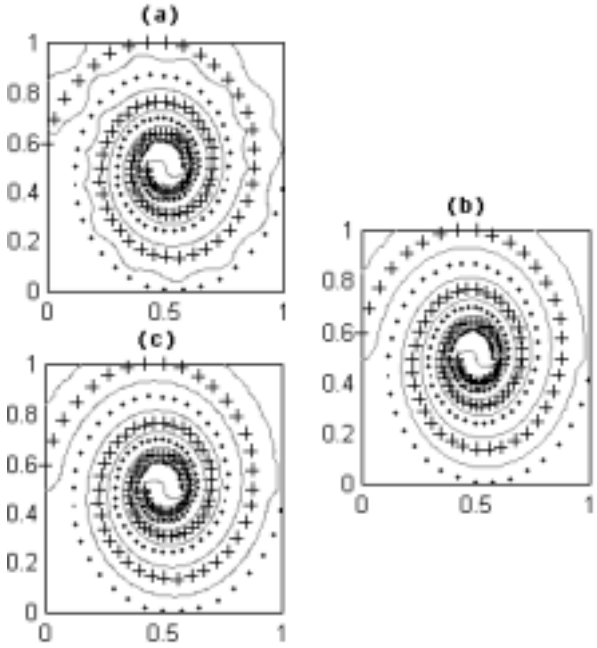

Figure 4. The hyperplanes of the kernel perceptron (a), LMKPA (b) and SVM (c), of the two-spiral problem. Results show that LMKPA overperforms kernel perceptron, and is almost identical with SVM.

VI. CONCLUSION

The kernel-based methods are becoming a new family of nonlinear techniques in the machine learning area, and the large margin classifiers are believed to be more robust with respect to samples and parameters of classifiers. In this paper we proposed a large margin kernel pocket algorithm (LMKPA), which is a simple iterative algorithm that realized both these two ideas. It can deal with both the linear and nonlinearly separable and non-separable cases. Its objective is to maximize simultaneously both the number of correctly classified samples and the minmial distance between the hyperplane and the correctly classified samples.

For the linear cases, LMKPA find a solution with large margin and without misclassified samples, which is identical to SVM. For otherwise cases, the performance of LMKPA is also very close to that of SVM, but its algorithm is much simpler.

REFERENCES

[1] F. Rosenblatt. The perceptron: probabilistic model for information storage and organization in the brain. Psychological Review, 65, 1958.

[2] R. O. Duda and P. E. Hart. Pattern Classification and Scene Analysis. John Wiley & Sons, New York, 1973.

[3] S. I. Gallant. Neural Networks learning and expert systems. MIT Press, Cambridge, MA , 1993.

[4] S. I. Gallant. Optimal linear discriminant. Proc. Eighth Int. Conf. Pattern Recognition (Paris, France), 849-853, 1986.

[5] S. I. Gallant. Perceptron-based learning algorithm. IEEE Transactions on Neural Networks, 1(2), 179-191, 1990.
[6] M. Muselli. On convergence properties of pocket algorithm. IEEE Transactions on Neural Networks, 8(3), 623-629, 1997.
[7] C. Cortes and V. N. Vapnik. Support Vector Networks. Machine Learning, 20(3), 273-297, 1995.
[8] V. N. Vapnik. Statistical Learning Theory. Wiley, New York, 1998.
[9] V. N. Vapnik. The Nature of Statistical Learning Theory (2nd ed.). Springer-Verlag, New York, 1999.
[10] A. J. Smola, P. Bartlett, B. Scholkopf, and C. Schuurmans (editors). Advances in Large Margin Classifiers. MIT Press, 2000.
[11] S. Mika, G. Ratsch, et al. Fisher discriminant analysis with kernels. Neural Networks for Signal Processing IX, 41-48. IEEE Press, New York, 1999.
[12] J. A. K. Suykens and J. Vandewalle. Least squares sup-port vector machines. Neural Processing Letters, 9, 293-300, 1999.
[13] I. Guyon and D. G. Stork. Linear discriminant and sup-port vector classifiers. In A. J. Smola, P. Bartlett, et al. (editors). Advances in Large Margin Classifiers. MIT Press, Cambridge, MA, 2000.
[14] J. XU, X. Zhang and Y. Li. Kernel perceptron algorithm. Technical Report, Department of Automation, Tsinghua University, 2000.
[15] J. T. Tou and R. C. Gonzadez. Pattern Recognition Principles. Addison-Wesly, Reading, 1974.
[16] T. Friess, N. Cristianini and C. Campbell. The kernel adatron: A fast and simple learning procedure for sup-port vector machines. Machine Learning: Proceedings of the Fifteenth International Conference, 1998.
[17] Y. Freund and R Schapire. Large margin classification using the perceptron algorithm. Machine Learning, 37, 277-296, 1999.
[18] A. Kowalczyk. Maximal margin perceptron. In A. J. Smola, P. Bartlett, et al. (editors). Advances in Large Margin Classifiers. MIT Press, Cambridge, MA, 2000.
[19] S. Theodoridis and K. Koutroumbas. Pattern Recognition. Academic Press, San Diego, 1999.
[20] N. Cristianini and J. S. Taylor. An introduction to sup-port vector machines and other kernel-based learning methods. Cambridge University Press, UK, 2000.
[21] H. Glucksman. On the improvement of a linear separation by extending the adaptive process with a stricter criterion. IEEE Transactions on Electronic Computers, 16(6), 941-944, 1966
[22] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2), 1-43, 1998.