Int. J. Appl. Math. Comput. Sci., 2004, Vol. 14, No. 1, 53–61

KERNEL HO-KASHYAP CLASSIFIER WITH GENERALIZATION CONTROL

JACEK ŁĘSKI[*]

[*] Institute of Electronics
Silesian University of Technology
ul. Akademicka 16, 44–100 Gliwice, Poland
e-mail: jl@boss.iele.polsl.gliwice.pl

This paper introduces a new classifier design method based on a kernel extension of the classical Ho-Kashyap procedure. The proposed method uses an approximation of the absolute error rather than the squared error to design a classifier, which leads to robustness against outliers and a better approximation of the misclassification error. Additionally, easy control of the generalization ability is obtained using the structural risk minimization induction principle from statistical learning theory. Finally, examples are given to demonstrate the validity of the introduced method.

Keywords: kernel methods, classifier design, Ho-Kashyap classifier, generalization control, robust methods

1. Introduction

Kernel-based methods in machine learning are concerned with increasing the computational power of linear methods by mapping the data into a high-dimensional feature space. This field of study was developed in the early 1990s and has recently played an important role in many engineering fields, such as pattern recognition, approximation, modeling, character recognition, data mining (Boser *et al*., 1992; Müller *et al*., 2001; Schölkopf *et al*., 1998, 1999). The following should be enumerated as examples of powerful kernel-based methods: support vector machines (Boser *et al*.,1992; Vapnik, 1995), kernel principal component analysis (Schölkopf *et al*., 1998, 1999a), kernel Fisher discriminant (Mika *et al*., 1999; Baudat and Anouar, 2000). A kernel classifier design method is of special interest in the paper. It implements the idea of mapping the input vectors $\mathbf{x}$ into a high (possibly infinite) dimensional feature space $\mathcal{F}$ through some nonlinear mapping. In this space, a linear separating hyperplane is constructed (Boser *et al*.,1992; Vapnik, 1995). Two problems arise in the above approach, a conceptual and a technical one (Vapnik, 1998; Müller *et al*., 2001): (i) How to find a separating hyperplane that generalizes well? The dimension of the feature space is huge and it is known from statistics that as for a function of the space dimensionality, we need exponentially many patterns to sample this space properly. (ii) How do we treat such a high-dimensional space computationally? For example, to construct a third-degree polynomial in a 50-dimensional space, hyperplanes in a 22100-dimensional space must be constructed!

Some very useful tools for solving the first problem (conceptual) are offered by statistical learning theory (the Vapnik-Chervonenkis (VC) theory). Let the expected risk (expected misclassification error rate) be denoted by $R(\alpha)$, where α denotes a parameter vector of a classifier. The empirical risk (misclassification error rate on a training set) is denoted by $R_{\text{emp}}(\alpha)$. Using the above notions, the following theorem can be proved (Vapnik, 1998; 1999): With probability $1-\eta$, the following bound holds:

$$R(\alpha) \le R_{\text{emp}}(\alpha) + \sqrt{\frac{h_{VC}(\log(2N/h_{VC})+1)-\log(\eta/4)}{N}}, \quad (1)$$

where N is the cardinality of the training set and h_{VC} is the Vapnik-Chervonenkis (VC) dimension that is a measure of the complexity of the set of functions from which the classifier is selected. The minimization of $R(\alpha)$ requires simultaneous minimization of both the terms on the right-hand side of (1), the first depending on the empirical risk $R_{\text{emp}}(\alpha)$ and the other on the VC dimension. The VC dimension for a set of functions is defined as the maximum number of data (from the training set) that can be shattered by these functions.

The most important issue in the VC theory is the Structural Risk Minimization (SRM) induction principle. Note that the second term on the right-hand side of (1) depends on the chosen class of functions, whereas the first term depends on the particular function chosen in the process of training. Let the set of functions Ψ used be composed of the nested subsets Ψ_k, $\Psi_1 \subset \Psi_2 \subset \cdots \subset \Psi_n$, such that $h_{VC}(\Psi_1) < h_{VC}(\Psi_2) < \cdots < h_{VC}(\Psi_n)$. The

SRM principle suggests that for a given training set we choose a subset Ψ_k and a particular function from Ψ_k for which the sum of terms on the right-hand side of (1) is minimal. In other words, the SRM principle suggests a tradeoff between the quality of the classification on the training set and the complexity of the classifier. So, we should select a classifier with the smallest VC dimension and the smallest misclassification error on the training set to achieve a good generalization capability. If the second term in (1) is large, then we could minimize the empirical risk down to zero, but the error rate on the test set might be important. In this case, the so-called overfitting (or overtraining) effect occurs.

From the above we see that the generalization ability is influenced by the complexity of the classifier rather than by the dimensionality of its input space. Thus, a classifier generalizes well when in a high-dimensional feature space it is chosen from a simple class of functions, for example, a linear class.

To solve the second problem (technical), a highly effective trick for computing the scalar product in the feature spaces is used (Vapnik, 1998; Müller *et al*., 2001). For certain feature spaces, using a kernel function instead of a scalar product in the original space corresponds to mapping this space into a scalar product in the feature space.

The support vector classifiers (SVC) inspired by VC theory try to find separation hyperplanes such that the expected misclassification error rate is minimized. The construction of the SVC leads to a quadratic programming (QP) problem with bound constraints and one linear equality constraint (Vapnik, 1998). In the literature, there are many classifiers, including the following kinds: statistical, linear discriminant, K-nearest neighbor, neural network, classification tree, and many more (Duda and Hart, 1973; Tou and Gonzalez, 1974; Ripley, 1996; Webb, 1999). But linear classifiers are of special interest, due to their simplicity and easy expansibility to nonlinear classifiers. One of the most powerful classical methods of designing linear classifiers is the least mean-squared error procedure with Ho-Kashyap modification (Ho and Kashyap, 1965, 1966). Two main disadvantages of this approach are: (i) the use of the quadratic loss function that leads to a poor approximation of the misclassification error rate, as well as to a non-robust method, (ii) the inability to control the VC dimension of the designed classifier.

The goal of this work is to introduce a kernel extension of the classical Ho-Kashyap procedure. This new method uses an approximation to the absolute loss function, resulting in robustness to outliers and a better approximation to the misclassification error. Additionally, this method controls the VC dimension of the designed classifier.

In the previous work (Łęski, 2003a), a linear Ho-Kashyap classifier with generalization control was introduced. Its nonlinear extension based on fuzzy if-then rules was shown in (Łęski, 2003b). The generalization of the above approach to the so-called ε-margin classifier using both local and global learning was presented in (Łęski, 2004). The remainder of this work is concerned with the kernel extension of the Ho-Kashyap classifier to a two-class problem. However, the proposed method can be easily generalized to multi-class problems using the class-rest or the class-class methodology (Tou and Gonzalez, 1974).

This paper is organized as follows: Section 2 presents a reformulation in scalar product space of the traditional Ho-Kashyap classifier design procedure with generalization control and approximation to the absolute loss function. Section 3 recalls nonlinear algorithms in kernel feature space and presents a nonlinear extension of the Ho-Kashyap procedure with generalization control. Simulation results and discussion for the classification of simple synthetic two-dimensional data and real-world high-dimensional data are shown in Section 4. Finally, conclusions are drawn in Section 5.

2. Ho-Kashyap Classifier in Scalar Product Space

A classifier is designed on the basis of a set of data called the training set, $\mathrm{Tr}^{(N)} = \{(\mathbf{x}_1, y_1),\ (\mathbf{x}_2, y_2),\ \cdots,\ (\mathbf{x}_N, y_N)\}$, where N is the data cardinality, and each independent datum (pattern) $\mathbf{x}_i \in \mathbb{R}^t$ has a corresponding dependent datum $y_i \in \{+1, -1\}$, which indicates the assignment to one of two classes, ω_1 or ω_2,

$$y_i = \begin{cases} +1, & \mathbf{x}_i \in \omega_1, \\ -1, & \mathbf{x}_i \in \omega_2. \end{cases} \tag{2}$$

We seek a weight vector $\mathbf{w} \in \mathbb{R}^t$ and a bias $w_0 \in \mathbb{R}$ such that

$$d(\mathbf{x}_i) \triangleq \mathbf{w}^\top \mathbf{x}_i + w_0 \begin{cases} > 0, & \mathbf{x}_i \in \omega_1, \\ < 0, & \mathbf{x}_i \in \omega_2, \end{cases} \tag{3}$$

where $d(\mathbf{x}_i)$ is called the linear discrimination (or decision) function.

If the conditions (3) are satisfied for all members of the training set, then the data are said to be linearly separable. For overlapping classes, it is impossible to find a weight vector $\mathbf{w}$ such that the conditions (3) are satisfied for all data. If we multiply by -1 all patterns of the training set that are members of the ω_2 class, then (3) can be rewritten in the form $y_i(\mathbf{w}^\top \mathbf{x}_i + w_0) > 0$ for $i = 1, 2, \ldots, N$. To improve the generalization ability of a classifier, a margin of separation ε is introduced (Vapnik, 1998): $y_i(\mathbf{w}^\top\mathbf{x}_i + w_0) \geq \varepsilon$ for $i = 1, 2, \dots, N$. Indeed, these inequalities are invariant under a positive scaling of $\mathbf{w}$, w_0 and ε. Thus, we can define a canonical hyperplane such that $y_i(\mathbf{w}^\top\mathbf{x}_i + w_0) \geq 1$. For the points closest to the canonical hyperplane, we have $y_i(\mathbf{w}_i^\top\mathbf{x} + w_0) = 1$. Let $\mathbf{x}_p$ and $\mathbf{x}_q$ be the closest points on the opposite sides of the separating hyperplane: $\mathbf{w}^\top\mathbf{x}_p + w_0 = 1$ and $\mathbf{w}^\top\mathbf{x}_q + w_0 = -1$. The margin is defined as the perpendicular distance between the hyperplanes through the closest points. The normal vector to the separating hyperplane is $\mathbf{w}/\|\mathbf{w}\|$. Thus, the margin is given by the projection of $\mathbf{x}_p - \mathbf{x}_q$ onto this vector and it is equal to $\mathcal{M} = 2/\|\mathbf{w}\|$. In this case, the VC dimension h_{VC} is bounded according to (Vapnik, 1995): $h_{VC} < R^2/\mathcal{M}^2 + 1$, where R is the diameter of the smallest ball around the data. Hence, if we bound the margin from below, we can control the VC-dimension of the class of separating hyperplanes.

Let us take into account a nested structure of the separating hyperplanes

$$\Psi_k = \{\mathbf{w}^\top\mathbf{x} + w_0 : \|\mathbf{w}\|^2 < a_k\}, \quad k = 1, 2, \dots,$$

where $a_k < a_{k+1}$.

In accordance with the SRM principle, a good generalization ability can be obtained by selecting the structure Ψ_j and a particular function from it with the smallest empirical misclassification error and the smallest VC dimension.

Let $\mathbf{X}_1$ be an $(\ell_1 \times t)$-dimensional matrix

$$\mathbf{X}_1 \triangleq \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_{\ell_1}^\top \end{bmatrix},$$

where $\mathbf{x}_1, \dots, \mathbf{x}_{\ell_1}$ are patterns from the ω_1 class, and let $\mathbf{X}_2$ be an $(\ell_2 \times t)$-dimensional matrix

$$\mathbf{X}_2 \triangleq \begin{bmatrix} \mathbf{x}_{\ell_1+1}^\top \\ \mathbf{x}_{\ell_1+2}^\top \\ \vdots \\ \mathbf{x}_{\ell_1+\ell_2}^\top \end{bmatrix},$$

where $\mathbf{x}_{\ell_1+1}, \dots, \mathbf{x}_{\ell_1+\ell_2}$ ($\ell_1 + \ell_2 = N$) are patterns from the ω_2 class. Note that for notational simplicity it is assumed that the patterns are ordered according to their class membership, that is, the first ℓ_1 patterns belong to the class ω_1.

Indeed, we seek a weight vector $\mathbf{w}$ and a bias w_0 such that

$$\begin{cases} \mathbf{X}_1\mathbf{w} + w_0\mathbf{1}_{\ell_1\times 1} \geq \mathbf{1}_{\ell_1\times 1}, \\ -\mathbf{X}_2\mathbf{w} - w_0\mathbf{1}_{\ell_2\times 1} \geq \mathbf{1}_{\ell_2\times 1}, \end{cases} \tag{4}$$

where $\mathbf{1}_{\ell_1\times 1}$ denotes the vector of dimension $\ell_1 \times 1$ with all entries equal to 1. To obtain a solution in scalar product space, we assume that the vector $\mathbf{w}$ is the following linear combination of all patterns: $\mathbf{w} = [\mathbf{X}_1^\top, -\mathbf{X}_2^\top]\mathbf{\Gamma}$, where $\mathbf{\Gamma} = [\gamma_1, \gamma_2, \cdots, \gamma_N]^\top$. Defining an $(N \times N)$-dimensional matrix

$$\mathcal{K} = \begin{bmatrix} \mathbf{X}_1\mathbf{X}_1^\top & -\mathbf{X}_1\mathbf{X}_2^\top \\ -\mathbf{X}_2\mathbf{X}_1^\top & \mathbf{X}_2\mathbf{X}_2^\top \end{bmatrix} = [y_i y_j \mathbf{x}_i^\top \mathbf{x}_j]_{i,j=1}^N, \tag{5}$$

and a vector $\boldsymbol{\theta}_{\ell_1,\ell_2} = [\mathbf{1}_{\ell_1\times 1}^\top, -\mathbf{1}_{\ell_2\times 1}^\top]^\top$, the inequalities (4) take the following form:

$$\mathcal{K}\mathbf{\Gamma} + w_0\boldsymbol{\theta}_{\ell_1,\ell_2} - \mathbf{1}_{N\times 1} \geq \mathbf{0}_{N\times 1}.$$

In order to solve the above system of inequalities, it is replaced by the linear system of equations $\mathcal{K}\mathbf{\Gamma} + w_0\boldsymbol{\theta}_{\ell_1,\ell_2} - \mathbf{1}_{N\times 1} = \mathbf{b}$, $\mathbf{b} \geq \mathbf{0}_{N\times 1}$. We define the error vector as $\mathbf{e} = \mathcal{K}\mathbf{\Gamma} + w_0\boldsymbol{\theta}_{\ell_1,\ell_2} - \mathbf{1}_{N\times 1} - \mathbf{b}$. If the p-th component of $\mathbf{e}$ is non-negative, then the p-th pattern falls on the right-hand side of the separation hyperplane, and by increasing the respective component of $\mathbf{b}$ (b_p), e_p can be reduced to zero. If the p-th component of $\mathbf{e}$ is negative, then the p-th pattern falls on the wrong side of the separation hyperplane, and it is impossible to prevent the condition $b_p \geq 0$ by decreasing b_p. Thus, the misclassification error may be written in the form

$$I = \sum_{i=1}^{N} \mathcal{H}(-e_i), \tag{6}$$

where $\mathcal{H}(\cdot)$ denotes the unit step pseudo-function, $\mathcal{H}(e_i) = 1$, for $e_i > 0$, and zero otherwise. We should minimize the criterion (6), but due to its non-convexity this optimization problem is NP-complete (Haykin, 1999).

To make this optimization problem tractable, we approximate (6) by a convex one

$$I = \sum_{i=1}^{N} |e_i| \tag{7}$$

or

$$I = \sum_{i=1}^{N} (e_i)^2. \tag{8}$$

The above approximations are possible due to the fact that the positive value of the error can be reduced to zero by increasing the respective components of $\mathbf{b}$.

In real life applications, data from the training set are corrupted by noise and outliers. Thus, classifier design methods need to be robust. According to Huber (Huber, 1981), a robust method should have the following properties: (i) it should have a reasonably good accuracy at the assumed model, (ii) small deviations from the model assumptions should impair the performance only by a small amount, (iii) larger deviations from the model assumptions should not cause a catastrophe. In the literature there are many robust loss functions (Huber, 1981). In this work, due to its simplicity, the absolute error loss function is of special interest.

The criterion (7) is a better approximation of (6) and additionally leads to a robust method, but for a mathematical reason, that is, the simplicity of the solution, we start from the criterion (8).

Now, we seek $\mathbf{\Gamma}$, $\mathbf{b}$ and w_0 by the following minimization:

$$\min_{\substack{\mathbf{\Gamma}\in\mathbb{R}^N \\ \mathbf{b}>\mathbf{0}_{N\times 1} \\ w_0\in\mathbb{R}}} I(\mathbf{\Gamma},\mathbf{b},w_0) \triangleq (\mathcal{K}\mathbf{\Gamma}+w_0\boldsymbol{\theta}_{\ell_1,\ell_2}-\mathbf{1}_{N\times 1}-\mathbf{b})^\top \times\mathbf{D}(\mathcal{K}\mathbf{\Gamma}+w_0\boldsymbol{\theta}_{\ell_1,\ell_2}-\mathbf{1}_{N\times 1}-\mathbf{b})+\tau\mathbf{\Gamma}^\top\mathcal{K}\mathbf{\Gamma}, \quad (9)$$

where $\mathbf{D}=\mathrm{diag}(d_1,d_2,\ldots,d_N)$ and d_i is the weight corresponding to the i-th pattern that can be interpreted as the reliability attached to this pattern. The criterion function (9) is the squared error weighted by coefficients d_i with the second term related to the minimization of the Vapnik-Chervonenkis dimension (complexity) of the classifier. The parameter $\tau\geq 0$ controls the trade-off between the classifier complexity and the amount up to which the errors are tolerated.

Optimality conditions are obtained by differentiating (9) with respect to $\mathbf{\Gamma}$, $\mathbf{b}$, w_0 and equating the results to zero:

$$\begin{cases}\tau\mathcal{K}\mathbf{\Gamma}+\mathcal{K}^\top\mathbf{D}\mathcal{K}\mathbf{\Gamma}+w_0\mathcal{K}^\top\mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2}=\mathcal{K}^\top\mathbf{D}(\mathbf{1}_{N\times 1}+\mathbf{b}),\\ \boldsymbol{\theta}_{\ell_1,\ell_2}^\top\mathbf{D}\mathcal{K}\mathbf{\Gamma}+w_0\boldsymbol{\theta}_{\ell_1,\ell_2}^\top\mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2}=\boldsymbol{\theta}_{\ell_1,\ell_2}^\top\mathbf{D}(\mathbf{1}_{N\times 1}+\mathbf{b}),\\ \mathbf{e}=\mathcal{K}\mathbf{\Gamma}+w_0\boldsymbol{\theta}_{\ell_1,\ell_2}-\mathbf{1}_{N\times 1}-\mathbf{b}=\mathbf{0}_{N\times 1}.\end{cases} \quad (10)$$

When defining the matrix

$$\mathbf{\Xi}=\begin{bmatrix}\mathbf{D}\mathcal{K}+\tau\mathbf{I} & \mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2}\\ \boldsymbol{\theta}_{\ell_1,\ell_2}^\top\mathbf{D}\mathcal{K} & \boldsymbol{\theta}_{\ell_1,\ell_2}^\top\mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2}\end{bmatrix}$$

and taking into account that $\mathcal{K}$ is symmetric, from the first two equations of (10) we obtain

$$\begin{bmatrix}\mathbf{\Gamma}\\ w_0\end{bmatrix}=\mathbf{\Xi}^{-1}\begin{bmatrix}\mathbf{D}(\mathbf{1}_{N\times 1}+\mathbf{b})\\ \boldsymbol{\theta}_{\ell_1,\ell_2}^\top\mathbf{D}(\mathbf{1}_{N\times 1}+\mathbf{b})\end{bmatrix}. \quad (11)$$

From (11) we see that the vector $\mathbf{\Gamma}$ and the bias w_0 depend on $\mathbf{b}$. The vector $\mathbf{b}$ may be called a margin vector because its components determine the distance from patterns to the separating hyperplane. For fixed $\mathbf{\Gamma}$, if a pattern lies on the right side of the hyperplane, the corresponding margin can be increased to obtain a zero error. However, if a pattern lies on the wrong side of the hyperplane, then the error is negative, and we can decrease the error only by decreasing the corresponding margin value. To prevent the positivity of $\mathbf{b}$, we start with $\mathbf{b}\geq\mathbf{0}_{N\times 1}$ and refuse to decrease any of its components using an iterative algorithm proposed by (Ho and Kashyap, 1965, 1966). Now, this algorithm can be extended to our weighted squared error criterion with regularization. The vector $\mathbf{\Gamma}$ and the parameter w_0 are determined on the basis of (11), that is,

$$\begin{bmatrix}\mathbf{\Gamma}^{(k)}\\ w_0^{(k)}\end{bmatrix}=\begin{bmatrix}\mathbf{D}^{(k)}\mathcal{K}+\tau\mathbf{I} & \mathbf{D}^{(k)}\boldsymbol{\theta}_{\ell_1,\ell_2}\\ \boldsymbol{\theta}_{\ell_1,\ell_2}^\top\mathbf{D}^{(k)}\mathcal{K} & \boldsymbol{\theta}_{\ell_1,\ell_2}^\top\mathbf{D}^{(k)}\boldsymbol{\theta}_{\ell_1,\ell_2}\end{bmatrix}^{-1} \times\begin{bmatrix}\mathbf{D}^{(k)}(\mathbf{1}_{N\times 1}+\mathbf{b}^{(k)})\\ \boldsymbol{\theta}_{\ell_1,\ell_2}^\top\mathbf{D}^{(k)}(\mathbf{1}_{N\times 1}+\mathbf{b}^{(k)})\end{bmatrix}, \quad (12)$$

where the superscript (k) denotes the iteration index. The components of vector $\mathbf{b}$ are modified by the components of the error vector $\mathbf{e}$, but only in the case when it results in an increase in the components of $\mathbf{b}$; otherwise, the components of $\mathbf{b}$ remain unmodified,

$$\mathbf{b}^{(k+1)}=\mathbf{b}^{(k)}+\rho(\mathbf{e}^{(k)}+|\mathbf{e}^{(k)}|), \quad (13)$$

where $0<\rho<1$ is a parameter.

Now, we shall show a method of selecting parameters d_i. This method leads to an approximation of the minimum absolute misclassification error (the criterion (7)). This approximation is easy to obtain by taking $d_i=1/|e_i|$ for $i=1,2,\ldots,N$, where e_i is the i-th component of the error vector. However, the error vector depends on $\mathbf{\Gamma}$ and w_0. So, we use the vector $\mathbf{\Gamma}$ and the bias w_0 from the previous iteration. This procedure is based on the claim that sequential vectors $\mathbf{\Gamma}^{(k)}$ and $w_0^{(k)}$ differ imperceptibly near the optimum solution. The procedure of classifier design can be summarized in the following steps:

1. Fix $\tau\geq 0$, $0<\rho<1$ and $\mathbf{D}^{(1)}=\mathbf{I}$. Initialize $\mathbf{b}^{(1)}\geq\mathbf{0}_{N\times 1}$. Set the iteration index $k=1$.
2. Calculate vector $\mathbf{\Gamma}^{(k)}$ and bias $w_0^{(k)}$ for the k-th iteration using (12).
3. Set $\mathbf{e}^{(k)}=\mathcal{K}\mathbf{\Gamma}^{(k)}+w_0^{(k)}\boldsymbol{\theta}_{\ell_1,\ell_2}-\mathbf{1}_{N\times 1}-\mathbf{b}^{(k)}$.
4. Set $d_i^{(k)}=1/|e_i^{(k)}|$ for $i=1,2,\ldots,N$, $\mathbf{D}^{(k+1)}=\mathrm{diag}(d_1^{(k)},\ldots,d_N^{(k)})$.
5. Set $\mathbf{b}^{(k+1)}=\mathbf{b}^{(k)}+\rho(\mathbf{e}^{(k)}+|\mathbf{e}^{(k)}|)$.
6. If $\|\mathbf{b}^{(k+1)}-\mathbf{b}^{(k)}\|_2>\xi$, then $k=k+1$ and go to Step 2, otherwise stop.

Remark 1. The quantity ξ is a pre-set parameter. If Step 4 in this algorithm is omitted, then the squared error minimization procedure is obtained. In practice, the divide-by-zero error in Step 4 does not occur. This follows from the fact that some components of vector $\mathbf{e}$ tend to zero as $k \to \infty$. But in this case the convergence of (9) to the minimum is slow and the condition in Step 6 stops the algorithm. Appendix shows that for $0 < \rho < 1$ and any diagonal matrix $\mathbf{D}$, the above algorithm is convergent to a local minimum of (9).

3. Kernel Ho-Kashyap Classifier

In the previous section the Ho-Kashyap method of classifier design is reformulated in such a way that only scalar products are used. Thus, now it is easy to introduce its nonlinear version using the idea of kernels. In this section, kernel functions are first recalled and then a nonlinear version of the Ho-Kashyap method is introduced.

3.1. Kernel Functions

Let $\Phi : \mathbf{x} \in \mathbb{R}^t \longmapsto \Phi(\mathbf{x}) \in \mathcal{F}$ be a nonlinear transformation of the input vectors $\mathbf{x}$ into a feature space $\mathcal{F}$. Depending on this transformation the feature space may be high- or even infinite-dimensional. Let us recall a simple example given by Vapnik (1995). If $\mathbf{x} = [x_1, x_2]^\top$ and $\Phi(\mathbf{x}) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]^\top$, then the scalar product in the feature space yields $\Phi(\mathbf{x})^\top\Phi(\mathbf{x}') = [x_1^2, \sqrt{2}x_1x_2, x_2^2][x_1'^2, \sqrt{2}x_1'x_2', x_2'^2]^\top = ([x_1, x_2][x_1', x_2']^\top)^2 = (\mathbf{x}^\top\mathbf{x}')^2 \triangleq k(\mathbf{x}, \mathbf{x}')$. Thus, in order to compute scalar products in the feature space $\mathcal{F}$, we use a kernel representation k, without explicitly using the transformation Φ. It is a direct consequence of (Müller, 2001): every linear algorithm that only uses scalar products can be easily extended to a nonlinear version by using kernels.

Mercer's theorem of functional analysis gives the answer to the question which function k corresponds to a scalar product in some feature space $\mathcal{F}$ (Haykin, 1999): If k is the continuous kernel of a positive integral operator on a Hilbert space $L_2(\mathtt{C})$ on a compact set $\mathtt{C} \subset \mathbb{R}^t$, that is,

$$\int_{\mathtt{C}} k(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') \,\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{x}' \geq 0 \quad \text{for all } f \in L_2(\mathtt{C}),$$

then k can be expanded into a uniformly convergent series of its orthogonal eigenfunctions $\{\psi_i\}$

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{N_{\mathcal{F}}} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{x}'),$$

where $N_{\mathcal{F}} \leq \infty$, $\lambda_i > 0$ are eigenvalues.

In this case, the mapping $\Phi : \mathbf{x} \longmapsto [\sqrt{\lambda_1}\psi_1(\mathbf{x}), \sqrt{\lambda_2}\psi_2(\mathbf{x}), \dots, \sqrt{\lambda_{N_{\mathcal{F}}}}\psi_{N_{\mathcal{F}}}(\mathbf{x})]^\top$ is a transformation of $\mathbf{x}$ into the feature space $\mathcal{F}$ such that k represents the scalar product $\Phi(\mathbf{x})^\top\Phi(\mathbf{x}') = k(\mathbf{x}, \mathbf{x}')$. We shall call Φ a feature map associated with (or induced by) kernel k. A short list of commonly used kernel functions is given in Table 1.

Table 1. Commonly used kernel functions.

Name	Form of $k(\mathbf{x}, \mathbf{x}')$
Polynomial	$(\alpha\mathbf{x}^\top\mathbf{x}' + 1)^d$, $d \in \mathbb{N}$, $\alpha \in \mathbb{R}$
Gaussian	$\exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2)$, $\gamma \in \mathbb{R}_+$
Sigmoidal	$\tanh(\gamma(\mathbf{x}^\top\mathbf{x}') + \alpha)$, $\gamma \in \mathbb{R}$, $\alpha \in \mathbb{R}$
Multiquadratic	$\sqrt{\|\mathbf{x} - \mathbf{x}'\|^2 + \alpha^2}$, $\alpha \in \mathbb{R}_+$
Inverse Multiquadratic	$\dfrac{1}{\sqrt{\|\mathbf{x} - \mathbf{x}'\|^2 + \alpha^2}}$, $\alpha \in \mathbb{R}_+$

3.2. Nonlinear Version of the Ho-Kashyap Classifier

Now, our goal is to construct a linear Ho-Kashyap classifier in the feature space $\mathcal{F}$. Equivalently, it means that we obtain a nonlinear Ho-Kashyap classifier in the original data space $\mathbf{x}$. If, instead of using the scalar product in $\mathbf{x}$ in (5), the kernel function is used,

$$\begin{aligned} \mathcal{K} &= \left[y_i y_j \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)\right]_{i,j=1}^N \\ &= \left[y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)\right]_{i,j=1}^N, \end{aligned} \tag{14}$$

then we obtain a linear classifier in the feature space $\mathcal{F}$, i.e., is a nonlinear one in the original input space. Thus, a nonlinear version of the Ho-Kashyap classifier is obtained by replacing matrix $\mathcal{K}$ in the algorithm from the previous section by the one given in (14). Now, we can represent the decision function of the classifier for an input pattern $\mathbf{x}$ as

$$\begin{aligned} d(\mathbf{x}) &= \operatorname{sign}\left(\sum_{i=1}^N y_i \gamma_i \Phi(\mathbf{x})^\top \Phi(\mathbf{x}_i) + w_0\right) \\ &= \operatorname{sign}\left(\sum_{i=1}^N y_i \gamma_i k(\mathbf{x}, \mathbf{x}_i) + w_0\right), \end{aligned}$$

where $\{\mathbf{x}_i\}_{i=1}^N$ denotes pattern from the training set, $\{y_i\}_{i=1}^N$ stands for pattern indicators to one of two classes, ω_1 or ω_2, and $\{\gamma_i\}_{i=1}^N$, w_0 are parameters of the classifier obtained in the training process.

4. Numerical Experiments and Discussion

In all experiments $\mathbf{b}^{(1)} = 10^{-6}\ \mathbf{1}_{N \times 1}$ was used. The iterations were stopped as soon as the Euclidean norm in a successive pair of $\mathbf{b}$ vectors was less than 10^{-4}. All computations were run on a Pentium IV

1.6 GHz computer running Windows NT4 and MATLAB environment. Benchmark databases were obtained via the Internet, cf. *http://ida.first.gmd.de/~raetsch/data*, *ftp://markov.stats.ox.ac.uk/pub/PRNN*, as well as the site *http://www.ics.uci.edu/~mlearn*.

4.1. Simple Synthetic Two-Dimensional Data

The purpose of this experiment was to compare the proposed method of classifier design with the support vector classifier and other classical classifiers. The simulations were performed for data generated by Ripley (1996). These data consist of patterns having two features and assigned to two classes. Each class has a bimodal distribution obtained as a mixture of two normal distributions. The class distribution was chosen to allow the best-possible error rate of about 8%. The training set consists of 250 patterns (125 patterns belong to each class), and the testing set consists of 1000 patterns (500 patterns belong to each class).

The parameter τ was in the range from 0 to 10, and the parameter ρ was equal to 0.99. For each value of τ after the training stage (a classifier design on the training set), the generalization ability of the classifier was determined as the misclassification error rate on the test set. The following kernel functions were used: Gaussian, sigmoidal and polynomial of orders 2, 3, 4. Table 2 shows the minimal error rate determined on the testing set for each kernel function.

Table 2. Simulation results for Ripley's two-class problem.

Kernel	Minimal error rate	Parameters
Gaussian	8.6%	$\tau = 2.6, \gamma = 1.7$
Sigmoidal	10.5%	$\tau = 0.1, \gamma = 0.1,$ $\alpha = 1$
Polynomial order 2	9.4%	$\tau = 0.1, \alpha = 0.5$
Polynomial order 3	9.4%	$\tau = 0.3, \alpha = 0.5$
Polynomial order 4	9.3%	$\tau = 0.3, \alpha = 0.5$

The best generalization, equal to 8.6%, is obtained for $\tau = 2.6$ and the Gaussian kernel ($\gamma = 1.7$). The decision function of the best generalizing classifier is presented with the testing set in Fig.1. This error rate is superior to the support vector machine (10.6%) and the relevance vector machine (9.3%) (Tipping, 2001). The nearest-prototype classifier with deterministic annealing optimization (Miller *et al.*, 1996) leads to the error rate equal to 8.6% for 12 prototypes, and the neuro-fuzzy classifier (Czogała and Łęski, 2000) leads to the error rate 8.8% for 2 fuzzy if-then rules.

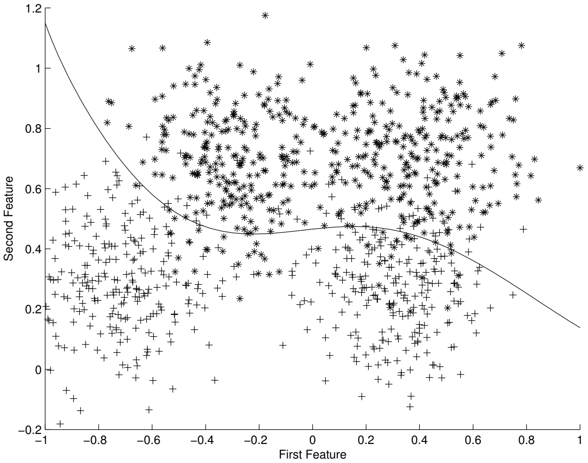

Fig. 1. Testing set for Ripley's two-class problem and the classifier line with the best generalization ability.

4.2. Real High-Dimensional Data

The main goal of these experiments was to examine the usefulness of the proposed method in the classification of real-world high-dimensional data. In order to investigate the performance of the proposed method, it is compared with the following methods: regularized AdaBoost, support vector machine and kernel Fisher discriminant, on 8 standard benchmarking datasets from the IDA repository (*http://ida.first.gmd.de/~raetsch/data*). For each dataset this repository includes 100 predefined splits into training and testing samples, and simulation results for several kernel-based and boosting methods with a summary including means and standard deviations of average misclassification error rates on the test sets. Some of these results are recalled in Table 3. In this table, all results are presented in percentages in the columns with the following abbreviations: AdaBoost — regularized AdaBoost (Rätsch *et al.*, 2001), SVM — Support Vector Machine, KFD — Kernel Fisher Discriminant and KHK — Kernel Ho-Kashyap. The Gaussian kernel was used for the KHK method. For each dataset in Table 3 the best result is in bold face and the best but one is underlined. This table also includes the parameter value for which the best generalization is obtained. Several observations can be made based on this table. First of all, it should be noted that for all databases the best generalization is obtained for a non-zero parameter τ. It must also be noted that the proposed method leads to the best generalization for banana, heart and titanic datasets. For breast cancer and thyroid datasets the best result but one is obtained. For the Flare solar dataset, the best result but one is obtained by KFD and KHK methods. However, the KHK method produces a smaller standard deviation. Finally, it can be noted that the proposed algorithm converges after tens of iterations and its

Tab. 3. Comparison between the regularized Adaboost (RAB), support vector machine (SVM), kernel Fisher discriminant (KFD) and kernel Ho-Kashyap classifier (KHK) on benchmark datasets. The best result for each dataset is marked in bold face and the best but one is underlined. The results for RAB, SVM and KFD are taken from (*http://ida.first.gmd.de/~raetsch/data*).

Database	RAB	SVM	KFD	KHK	τ	γ
Banana	10.85 ± 0.42	11.53 ± 0.66	$\underline{10.75 \pm 0.45}$	$\mathbf{10.49 \pm 0.47}$	0.4	1.0
Breast Cancer	26.51 ± 4.47	26.04 ± 4.74	$\mathbf{24.77 \pm 4.63}$	$\underline{25.42 \pm 4.24}$	1.8	0.1
Diabetis	23.79 ± 1.80	$\underline{23.53 \pm 1.73}$	$\mathbf{23.21 \pm 1.63}$	23.85 ± 1.60	1.9	0.1
Flare Solar	34.20 ± 2.18	$\mathbf{32.43 \pm 1.82}$	33.16 ± 1.72	$\underline{33.16 \pm 1.46}$	4.5	0.3
German	24.34 ± 2.08	$\mathbf{23.61 \pm 2.07}$	23.71 ± 2.20	$\underline{23.68 \pm 2.19}$	0.9	0.05
Heart	16.47 ± 3.51	$\underline{15.95 \pm 3.26}$	16.14 ± 3.39	$\mathbf{15.62 \pm 3.53}$	1.8	0.01
Thyroid	4.55 ± 2.19	4.80 ± 2.19	$\mathbf{4.20 \pm 2.07}$	$\underline{4.26 \pm 1.95}$	3.4	0.8
Titanic	22.64 ± 1.20	$\underline{22.42 \pm 1.02}$	23.25 ± 2.05	$\mathbf{22.36 \pm 1.01}$	0.1	0.6

running time is several times shorter compared with the support vector machine. The Support Machine Toolbox by Steve Gunn was chosen as the SVM implementation. The Matlab code of this toolbox is available at *http://www.isis.ecs.sofon.ac.uk/resources/svminfo/download.php*. For the diabetis dataset, the running time of the support vector machine was about 80 times larger compared with the KHK method.

5. Conclusions

A new classifier design method is introduced. This method is a kernel extension of the classical Ho-Kashyap methodology which uses an approximation of the absolute loss function rather than the quadratic one. This results in robustness to outliers and a better approximation of the misclassification error. Additionally, the proposed method minimizes the Vapnik-Chervonenkis dimension that results in an easy control of the generalization ability of the classifier. Numerical examples are given to illustrate the validity of the presented method. These examples show that the proposed method has excellent generalization performance on real-world high-dimensional data. A comparison of the generalization ability of the kernel Ho-Kashyap method with the state-of-the-art classifiers, such as the regularized Adaboost, the support vector machine and the kernel Fisher discriminant shows that the kernel Ho-Kashyap classifier outperforms other methods on most datasets.

From the computational point of view the inverse of a matrix is needed in the kernel Ho-Kashyap method instead of a quadratic programming problem in the support vector machine and an eigenvector/eigenvalue problem in the kernel Fisher discriminant. This must be viewed as an advantage because calculating the inverse of a matrix is probably one of the best numerical methods established. The kernel Ho-Kashyap method shows that not only a sparse but also a dense classifier has a good generalization ability.

References

Baudat G. and Anouar F. (2000): *Generalized discriminant analysis using a kernel approach*. — Neural Comput., Vol. 12, No. 10, pp. 2385–2404.

Boser B.E., Guyon I.M. and Vapnik V. (1992): *A training algorithm for optimal margin classifiers*. — Proc. 5th Ann. ACM Workshop *Computational Learning Theory*, Pittsburgh, USA, pp. 144–152.

Czogała E. and Łęski J.M. (2000): *Fuzzy and Neuro-Fuzzy Intelligent Systems*. — Heidelberg: Physica-Verlag.

Duda R.O. and Hart P.E. (1973): *Pattern Classification and Scene Analysis*. — New York: Wiley.

Gantmacher F.R. (1959): *The Theory of Matrices*. — New York: Chelsea Publ.

Haykin S. (1999): *Neural Networks. A Comprehensive Foundation*. — Upper Saddle River: Prentice-Hall.

Ho Y.-C. and Kashyap R.L. (1965): *An algorithm for linear inequalities and its applications*. — IEEE Trans. Elec. Comp., Vol. 14, No. 5, pp. 683–688.

Ho Y.-C. and Kashyap R.L. (1966): *A class of iterative procedures for linear inequalities*. — SIAM J. Control., Vol. 4, No. 2, pp. 112–115.

Huber P.J. (1981): *Robust Statistics*. — New York: Wiley.

Łęski J.M. (2003a): *Ho-Kashyap classifier with generalization control*. — Pattern Recogn. Lett., Vol. 24, No. 2, pp. 2281–2290.

Łęski J.M. (2003b): *Fuzzy if-then rule-based nonlinear classifier*. — Int. J. Appl. Math. Comput. Sci., Vol. 13, No. 2, pp. 101–109.

Łęski J.M. (2004): *An ε-margin nonlinear classifier based on if-then rules*. — IEEE Trans. Sys. Man Cybern. – Part B: Cybernet., Vol. 34, No. 1, pp. 68–76.

Mika S., Rätsch G., Weston J., Schölkopf B. and Müller K.-R. (1999): *Fisher discriminant analysis with kernels*, In: Neural Networks in Signal Processing IX (Y.H. Hu, J. Larsen, E. Wilson and S. Douglas, Eds.). — New York: IEEE Press, pp. 41–48.

Miller D., Rao A.V., Rose K. and Gersho A. (1996): *A global optimization technique for statistical classifier design*. — IEEE Trans. Signal Process., Vol. 44, No. 12, pp. 3108–3121.

Müller K.-R., Mika S., Rätsch G., Tsuda K. and Schölkopf B. (2001): *An introduction to kernel-based learning algorithms*. — IEEE Trans. Neural Netw., Vol. 12, No. 2, pp. 181–202.

Rätsch G., Onoda T. and Müller K.-R. (2001): *Soft margins for AdaBoost*. — Mach. Learn., Vol. 42, No. 3, pp. 287–320.

Ripley B.D. (1996): *Pattern Recognition and Neural Networks*. — Cambridge: Cambridge University Press.

Schölkopf B., Smola A.J. and Müller K.-R. (1998): *Nonlinear component analysis as a kernel eigenvalue problem*. — Neural Comput., Vol. 10, No. 6, pp. 1299–1319.

Schölkopf B., Burges C.J.C. and Smola A.J. (1999): *Advances in Kernel Methods – Support Vector Machine*. — Cambridge: MIT Press.

Schölkopf B., Mika S., Burges C.J.C., Knirsch P., Müller K.-R., Rätsch G. and Smola A.J. (1999a): *Input space vs. feature space in kernel-based methods*. — IEEE Trans. Neural Netw., Vol. 10, No. 5, pp. 1000–1017.

Tipping M.E. (2001): *Sparse Bayesian learning and the relevance vector machine*, — J. Mach. Learn. Res., Vol. 1, No. 2, pp. 211–244.

Tou J.T. and Gonzalez R.C. (1974): *Pattern Recognition Principles*. — London: Adison-Wesley.

Vapnik V. (1995): *The Nature of Statistical Learning Theory*. — New York: Springer-Verlag.

Vapnik V. (1998): *Statistical Learning Theory*. — New York: Wiley.

Vapnik V. (1999): *An Overview of Statistical Learning Theory*. — IEEE Trans. Neural Netw., Vol. 10, No. 5, pp. 988–999.

Webb A. (1999): *Statistical Pattern Recognition*. — London: Arnold.

Appendix

The first equation of (10) can be rewritten in the form $\mathcal{K}^\top \mathbf{D}\mathbf{e} = -\tau \mathcal{K}\boldsymbol{\Gamma}$. Thus, for $\tau > 0$ all elements of the error vector cannot be zero. This is true in both linearly separable and nonseparable cases. If we define

$$\boldsymbol{\Xi}^{-1} = \begin{bmatrix} \boldsymbol{\Delta}_{11} & \boldsymbol{\Delta}_{12} \\ \boldsymbol{\Delta}_{21} & \boldsymbol{\Delta}_{22} \end{bmatrix},$$

then the use of the extension principle[1] yields (Gantmacher, 1959):

$$\begin{aligned} \boldsymbol{\Delta}_{11} &= (\mathbf{D}\mathcal{K} + \tau\mathbf{I})^{-1} + \frac{1}{\beta'}(\mathbf{D}\mathcal{K} + \tau\mathbf{I})^{-1} \\ &\quad \times \mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2}\boldsymbol{\theta}_{\ell_1,\ell_2}^\top \mathbf{D}\mathcal{K}(\mathbf{D}\mathcal{K} + \tau\mathbf{I})^{-1}, \\ \boldsymbol{\Delta}_{12} &= -\frac{1}{\beta'}(\mathbf{D}\mathcal{K} + \tau\mathbf{I})^{-1}\mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2}, \\ \boldsymbol{\Delta}_{21} &= -\frac{1}{\beta'}\boldsymbol{\theta}_{\ell_1,\ell_2}^\top \mathbf{D}\mathcal{K}(\mathbf{D}\mathcal{K} + \tau\mathbf{I})^{-1}, \\ \boldsymbol{\Delta}_{22} &= \frac{1}{\beta'}, \end{aligned}$$

where

$$\beta' = \boldsymbol{\theta}_{\ell_1,\ell_2}^\top \mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2} - \boldsymbol{\theta}_{\ell_1,\ell_2}^\top \mathbf{D}\mathcal{K}(\mathbf{D}\mathcal{K} + \tau\mathbf{I})^{-1}\mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2}.$$

Write $\mathbf{e}_+^{(k)} \triangleq \mathbf{e}^{(k)} + |\mathbf{e}^{(k)}|$. Using (12) and (13), we obtain

$$\begin{aligned} \mathbf{e}^{(k+1)} &= \mathbf{e}^{(k)} + \rho(\mathcal{K}\boldsymbol{\Delta}_{11}\mathbf{D} + \mathcal{K}\boldsymbol{\Delta}_{12}\boldsymbol{\theta}_{\ell_1,\ell_2}^\top \mathbf{D} \\ &\quad + \boldsymbol{\theta}_{\ell_1,\ell_2}\boldsymbol{\Delta}_{21}\mathbf{D} + \boldsymbol{\theta}_{\ell_1,\ell_2}\boldsymbol{\Delta}_{22}\boldsymbol{\theta}_{\ell_1,\ell_2}^\top \mathbf{D} - \mathbf{I})\mathbf{e}_+^{(k)} \end{aligned}$$

and

$$\boldsymbol{\Gamma}^{(k+1)} = \boldsymbol{\Gamma}^{(k)} + \rho(\boldsymbol{\Delta}_{11}\mathbf{D} + \boldsymbol{\Delta}_{12}\boldsymbol{\theta}_{\ell_1,\ell_2}^\top \mathbf{D})\mathbf{e}_+^{(k)}.$$

From the second equation of (10) we obtain

$$\boldsymbol{\theta}_{\ell_1,\ell_2}^\top \mathbf{D}\mathbf{e}_+ = \mathbf{e}_+^\top \mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2} = 0.$$

Using the above result yields

$$\mathbf{e}^{(k+1)} = \mathbf{e}^{(k)} + \rho(\mathcal{K}\boldsymbol{\Delta}_{11}\mathbf{D} + \boldsymbol{\theta}_{\ell_1,\ell_2}\boldsymbol{\Delta}_{21}\mathbf{D} - \mathbf{I})\mathbf{e}_+^{(k)}$$

and

$$\boldsymbol{\Gamma}^{(k+1)} = \boldsymbol{\Gamma}^{(k)} + \rho\boldsymbol{\Delta}_{11}\mathbf{D}\mathbf{e}_+^{(k)}.$$

[1] The extension principle is formulated as follows: $\begin{bmatrix} \mathbf{Z} & u \\ v^\top & \beta \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{Z}^{-1} + \frac{\mathbf{Z}^{-1}uv^\top\mathbf{Z}^{-1}}{\beta'} & -\frac{\mathbf{Z}^{-1}u}{\beta'} \\ -\frac{v^\top\mathbf{Z}^{-1}}{\beta'} & \frac{1}{\beta'} \end{bmatrix}$, where $\beta' = \beta - v^\top\mathbf{Z}^{-1}u$. In our case, $\mathbf{Z} = \mathbf{D}\mathcal{K} + \tau\mathbf{I}$, $v^\top = \boldsymbol{\theta}_{\ell_1,\ell_2}^\top \mathbf{D}\,\mathcal{K}$, $u = \mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2}$ and $\beta = \boldsymbol{\theta}_{\ell_1,\ell_2}^\top \mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2}$.

Substitution of the above results in (9) gives

$$
\begin{aligned}
&I^{(k+1)}\\
&\quad = I^{(k)} + 2\rho\mathbf{e}^{(k)\top}\mathbf{D}(\mathcal{K}\boldsymbol{\Delta}_{11}\mathbf{D}+\boldsymbol{\theta}_{\ell_1,\ell_2}\boldsymbol{\Delta}_{21}\mathbf{D}-\mathbf{I})\mathbf{e}_+^{(k)}\\
&\qquad +\rho^2\mathbf{e}_+^{(k)\top}(\mathbf{D}\boldsymbol{\Delta}_{11}^\top\mathcal{K}+\mathbf{D}\boldsymbol{\Delta}_{21}^\top\boldsymbol{\theta}_{\ell_1,\ell_2}^\top-\mathbf{I})\\
&\qquad \times\mathbf{D}(\mathcal{K}\boldsymbol{\Delta}_{11}\mathbf{D}+\boldsymbol{\theta}_{\ell_1,\ell_2}\boldsymbol{\Delta}_{21}\mathbf{D}-\mathbf{I})\mathbf{e}_+^{(k)}\\
&\qquad +2\tau\rho\boldsymbol{\Gamma}^{(k)\top}\mathcal{K}\boldsymbol{\Delta}_{11}\mathbf{D}\mathbf{e}_+^{(k)}\\
&\qquad +\tau\rho^2\mathbf{e}_+^{(k)\top}\mathbf{D}\boldsymbol{\Delta}_{11}^\top\mathcal{K}\boldsymbol{\Delta}_{11}\mathbf{D}\mathbf{e}_+^{(k)}.
\end{aligned}
$$

From the first equation of (10) we have

$$\mathcal{K}^\top\mathbf{D}\mathbf{e} = -\tau\mathcal{K}\boldsymbol{\Gamma}.$$

Using the above results and the equality

$$
\begin{aligned}
&2\rho\mathbf{e}^{(k)\top}\mathbf{D}(\mathcal{K}\boldsymbol{\Delta}_{11}\mathbf{D}+\boldsymbol{\theta}_{\ell_1,\ell_2}\boldsymbol{\Delta}_{21}\mathbf{D}-\mathbf{I})\mathbf{e}_+^{(k)}\\
&\quad = \rho\mathbf{e}_+^{(k)\top}\mathbf{D}(\mathcal{K}\boldsymbol{\Delta}_{11}\mathbf{D}+\boldsymbol{\theta}_{\ell_1,\ell_2}\boldsymbol{\Delta}_{21}\mathbf{D}-\mathbf{I})\mathbf{e}_+^{(k)},
\end{aligned}
$$

after some simple algebra, we obtain

$$
\begin{aligned}
&I^{(k+1)} - I^{(k)}\\
&\quad = \rho(\rho-1)\mathbf{e}_+^{(k)\top}\mathbf{D}\mathbf{e}_+^{(k)}\\
&\qquad -\rho^2\mathbf{e}_+^{(k)\top}\mathbf{D}\mathcal{K}(\mathbf{D}\mathcal{K}+\tau\mathbf{I})^{-1}\mathbf{D}\mathbf{e}_+^{(k)}\\
&\qquad -\frac{\rho^2}{\beta'}\mathbf{e}_+^{(k)\top}\mathbf{D}(\mathbf{D}\mathcal{K}+\tau\mathbf{I})^{-\mathbf{1}}\\
&\qquad \times\mathcal{K}\mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2}\boldsymbol{\theta}_{\ell_1,\ell_2}^\top\mathbf{D}\mathcal{K}(\mathbf{D}\mathcal{K}+\tau\mathbf{I})^{-\mathbf{1}}\mathbf{D}\mathbf{e}_+^{(k)}.
\end{aligned}
$$

The matrices

$$\mathbf{D}\mathcal{K}(\mathbf{D}\mathcal{K}+\tau\mathbf{I})^{-1}\mathbf{D}$$

and

$$\mathbf{D}(\mathbf{D}\mathcal{K}+\tau\mathbf{I})^{-\mathbf{1}}\mathcal{K}\mathbf{D}\boldsymbol{\theta}_{\ell_1,\ell_2}\boldsymbol{\theta}_{\ell_1,\ell_2}^\top\mathbf{D}\mathcal{K}(\mathbf{D}\mathcal{K}+\tau\mathbf{I})^{-\mathbf{1}}\mathbf{D}$$

are symmetric and positive semidefinite. As a result, the second and third terms are negative or zero. For $0 < \rho < 1$ the first term is negative or zero. Thus, the sequence $I^{(1)}$, $I^{(2)}, \dots$ is monotonically decreasing. For both linearly separable and nonseparable cases, convergence requires that $\mathbf{e}_+^{(k)}$ tends to zero (no modification in (13)), while $\mathbf{e}^{(k)}$ is bounded away from zero, due to $\mathcal{K}^\top\mathbf{D}\mathbf{e} = -\tau\mathcal{K}\boldsymbol{\Gamma}$.

Received: 18 August 2003
Revised: 5 January 2004