A Bayesian Approach to Joint Feature Selection and Classifier Design

Balaji Krishnapuram, Alexander J. Hartemink, Lawrence Carin, *Fellow*, *IEEE*, and Mário A.T. Figueiredo, *Senior Member*, *IEEE*

Abstract—This paper adopts a Bayesian approach to simultaneously learn both an optimal nonlinear classifier and a subset of predictor variables (or features) that are most relevant to the classification task. The approach uses heavy-tailed priors to promote sparsity in the utilization of both basis functions and features; these priors act as regularizers for the likelihood function that rewards good classification on the training data. We derive an expectation-maximization (EM) algorithm to efficiently compute a *maximum a posteriori* (MAP) point estimate of the various parameters. The algorithm is an extension of recent state-of-the-art sparse Bayesian classifiers, which in turn can be seen as Bayesian counterparts of support vector machines. Experimental comparisons using kernel classifiers demonstrate both parsimonious feature selection and excellent classification accuracy on a range of synthetic and benchmark data sets.

Index Terms—Pattern recognition, statistical learning, feature selection, sparsity, support vector machines, relevance vector machines, sparse probit regression, automatic relevance determination, EM algorithm.

1 Introduction

1.1 Motivation

In binary supervised learning problems, the goal is to learn how to distinguish between examples from two classes (herein labeled, without loss of generality, as $y = 0$ and $y = 1$) on the basis of p observed predictor variables (also known as features) $\boldsymbol{x} = [x_1, x_2, \ldots, x_p]^T \in \mathbb{R}^p$. To achieve this goal, we are given a training set $D = \{(\boldsymbol{x}^{(i)}, y^{(i)}) : \boldsymbol{x}^{(i)} \in \mathbb{R}^p, y^{(i)} \in \{0,1\}\}_{i=1}^n$ with n labeled examples, where $y^{(i)}$ is the label associated with example $\boldsymbol{x}^{(i)}$. Given this training set D, two standard tasks are to learn a function that most accurately predicts the class of a new example (classifier design) and to identify a subset of the features that is most informative about the class distinction (feature selection). In this paper, we develop a *joint classifier and feature optimization* (JCFO) algorithm to accomplish both tasks simultaneously.[1]

One fundamental idea undergirding JCFO is the association of a nonnegative scaling factor θ_k with each feature x_k; these scaling factors are then estimated from the data, under an a priori preference for values that are either significantly large or otherwise exactly zero. Although this idea can be applied to a variety of other types of classifiers, we focus in this paper on probabilistic kernel classifiers of the form

$$P(y = 1|\boldsymbol{x}) = \Phi\left(\beta_0 + \sum_{i=1}^{n} \beta_i K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^{(i)})\right), \tag{1}$$

where $\Phi(z)$ is the Gaussian cumulative distribution function (also known as the *probit link* [12]),

$$\Phi(z) = \int_{-\infty}^{z} \mathcal{N}(x|0,1)\,dx = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp\left(\frac{-x^2}{2}\right) dx, \tag{2}$$

and $K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^{(i)})$ is a (possibly nonlinear) measure of similarity between the test example $\boldsymbol{x}$ and the training example $\boldsymbol{x}^{(i)}$; usually $K_{\boldsymbol{\theta}}(\cdot,\cdot)$ is called the *kernel function* [16]. In our context, the similarity between two examples depends upon the scaling factors associated with each of the features; we indicate the parameter dependence of the kernel function on the scaling factors by the use of the subscript $\boldsymbol{\theta}$. Two of the most popular kernels are easily parameterized to incorporate these scaling factors:

$$rth\ order\ polynomial: \quad K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^{(i)}) = \left(1 + \sum_{k=1}^{p} \theta_k x_k x_k^{(i)}\right)^r \tag{3}$$

$$Gaussian\ radial\ basis\ function\ (RBF): \quad K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^{(i)}) = \exp\left(-\sum_{k=1}^{p} \theta_k \left(x_k - x_k^{(i)}\right)^2\right). \tag{4}$$

From these two equations, we see that setting $\theta_k = 0$ is equivalent to removing the kth feature from the classifier entirely.

JCFO seeks sparsity in its use of both basis functions (sparsity in $\boldsymbol{\beta} = [\beta_0, \ldots, \beta_n]^T$) and features (sparsity in $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_p]^T$). This dual emphasis is founded on two observations. First, for kernel classification, sparsity in the use of basis functions is known to impact the *capacity* of the classifier, which controls its generalization performance [6]. Second, according to the well-known *curse of dimensionality*,

1. A preliminary conference version of this paper concentrating mostly on biological applications was presented in [10].

• *B. Krishnapuram and L. Carin are with the Department of Electrical Engineering, Duke University, Durham, NC 27708-0291. E-mail: {balaji, lcarin}@ee.duke.edu.*
• *A.J. Hartemink is with the Department of Computer Science, Duke University, Durham, NC 27708-0129. E-mail: amink@cs.duke.edu.*
• *M.A.T. Figueiredo is with the Insituto de Telecomunicações, Instituto Superior Técnico, 1049-001 Lisboa, Portugal. E-mail: mtf@lx.it.pt.*

Manuscript received 20 Aug. 2003; revised 2 Feb. 2004; accepted 5 Mar. 2004. Recommended for acceptance by L. Kuncheva.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0239-0803.

the difficulty of a learning task increases as the feature dimensionality grows relative to the number of training samples [3]; thus, sparsity in feature utilization is another important factor for increased robustness. Motivated by these observations, JCFO includes an a priori preference for sparsity in both sets of parameters, $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, as they are simultaneously estimated from the training set D.

1.2 Related Work

Except for the choice of the probit link, the classifier shown in (1) is similar to other kernel machines such as the *support vector machine* (SVM) [16] and the *relevance vector machine* (RVM) [18]. The SVM and RVM learning algorithms also seek sparsity in $\boldsymbol{\beta}$, but do not address the estimation of scaling factors $\boldsymbol{\theta}$.

Automatic relevance determination (ARD) [13] has been used to estimate scaling factors on features for several different types of classifiers, including neural networks [7], [13], RBF networks [8], and kernel classifiers [14], [20]. While algorithms that use ARD are designed to find scaling factors θ_k that are well-matched to the training data D, JCFO differs from these ARD methods in that it incorporates an explicit sparsity-promoting (heavy-tailed) prior. This results in many coefficient estimates that are *exactly* zero, as opposed to estimates that are close to but not exactly zero. This behavior is controlled by a tunable parameter of the prior; in practice, adjusting this parameter through cross-validation results in both strong feature selection and good classification accuracy in data sets with high feature dimensionality and many irrelevant features, like gene expression data. Moreover, even in benchmark data sets with low feature dimensionality, with few or no irrelevant features, JCFO is found to perform at least as well as conventional ARD methods, both in terms of classification accuracy and feature scale estimation properties.

Methods specifically tailored for feature selection in SVMs have also been proposed [5], [19]. In contrast with JCFO, those methods involve training optimal SVM classifiers over several feature subsets; selection is then performed by recursive feature elimination [5] or by adjusting the scaling factors using the gradient of a (loose) theoretical upper bound on the error rate [19]. In Section 4, we compare the classification performance of our JCFO algorithm against the best results for all of these methods, using benchmark data sets.

Finally, we should point out that JCFO is a generalization of the *sparse probit regression* (SPR) technique of [4] which only learns a sparse set of basis functions $\boldsymbol{\beta}$. In contrast, JCFO jointly learns a sparse set of basis functions $\boldsymbol{\beta}$ and a sparse set of features $\boldsymbol{\theta}$. If we chose the basis functions to be the features themselves (i.e., a nonkernelized linear hyperplane classifier), the roles of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ would merge and JCFO would be essentially equivalent to SPR.

1.3 Structure of the Remainder of the Paper

In Section 2, we discuss the sparsity-promoting priors used by JCFO. We derive an efficient expectation-maximization (EM) algorithm to jointly estimate $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ in Section 3. In Section 4, we report the results of various experiments comparing our JCFO algorithm to several state-of-the-art methods (e.g., SVM, RVM, and SPR) on synthetic data sets, high-dimensional gene expression data sets, and low-dimensional benchmark data sets. Section 5 concludes the paper with a discussion of the results and a survey of possible future research directions.

2 Sparsity-Promoting Priors

To encourage sparsity in the estimates of the parameter vectors $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, we adopt a Laplacian prior for each. It is known from earlier work that a Laplacian prior promotes sparsity, due to its equivalence to regularization with an l_1 norm penalty [4], [17]; formally, $p(\boldsymbol{\beta}|\gamma) \propto \exp(-\gamma||\boldsymbol{\beta}||_1)$, where $||\boldsymbol{\beta}||_1 = \sum_i |\beta_i|$ is the l_1 norm. Notice that, for small β_i, the difference between $p(0)$ and $p(\beta_i)$ is much larger for a Laplacian than for a Gaussian. As a result, using a Laplacian prior in a learning procedure that seeks to maximize the posterior density $p(\beta_i|D) \propto p(D|\beta_i)p(\beta_i)$ strongly favors values of β_i that are exactly 0 over values that are simply close to 0. More insight on the sparsity-promoting nature of the Laplacian prior can be found in [4], [17].

If we attempt to use a Laplacian prior directly, we will run into computational difficulties due to its nondifferentiability at the origin. To avoid this difficulty, we use an alternative hierarchical formulation which is equivalent to the Laplacian prior [4]. Let each β_i have a zero-mean Gaussian prior $p(\beta_i|\tau_i) = \mathcal{N}(\beta_i|0, \tau_i)$, with its own variance $\tau_i \geq 0$. Further, let all the variances τ_i be independently distributed according to a common exponential distribution (the hyperprior): $p(\tau_i|\gamma_1) = (\gamma_1/2)\exp(-\gamma_1\,\tau_i/2)$, for $\tau_i \geq 0$. With these assumptions in place, the effective prior can be obtained by marginalizing with respect to each τ_i,

$$p(\beta_i|\gamma_1) = \int_0^\infty p(\beta_i|\tau_i)\; p(\tau_i|\gamma_1)\; d\tau_i = \frac{\sqrt{\gamma_1}}{2}\exp(-\sqrt{\gamma_1}|\beta_i|), \quad (5)$$

showing that this hierarchical model is indeed equivalent to a Laplacian prior.

For each scaling coefficient θ_k, we adopt a similar sparsity-promoting prior, but with an important difference: We must ensure that any estimate of θ_k is nonnegative. To see why, observe that, in the kernels defined by (3) and (4), a negative value for θ_k would imply that when comparing two feature vectors, increasing similarity of the corresponding features would correspond to a reduced value of the kernel function. Though greater similarity of a particular feature in two different observations need not imply that the observations are *more* similar—e.g., if this feature is irrelevant—it should never imply that they are somehow *less* similar. Thus, although θ_k can and often will be exactly zero, negative values are disallowed. Accordingly, we consider a hierarchical model for θ_k similar to the one described above for β_i, but with the Gaussian prior replaced by a truncated Gaussian prior that explicitly forbids negative values

$$p(\theta_k|\rho_k) = \begin{cases} 2\,\mathcal{N}(\theta_k|0, \rho_k) & \text{if } \theta_k \geq 0 \\ 0 & \text{if } \theta_k < 0 \end{cases} \quad (6)$$

and an exponential hyperprior $p(\rho_k|\gamma_2) = (\gamma_2/2)\exp(-\gamma_2\,\rho_k/2)$, for $\rho_k \geq 0$. As above, the effective prior can be obtained by marginalizing with respect to ρ_k,

$$p(\theta_k|\gamma_2) = \int_0^\infty p(\theta_k|\rho_k)\, p(\rho_k|\gamma_2)\, d\rho_k = \begin{cases} \sqrt{\gamma_2}\, \exp(-\sqrt{\gamma_2}\theta_k) & \text{if } \theta_k \geq 0 \\ 0 & \text{if } \theta_k < 0, \end{cases} \tag{7}$$

showing that the effective prior on θ_k has the same shape as a Laplacian density, but limited to positive values of the parameter (i.e., it is an exponential density).

3 Map Parameter Estimation via EM

Given the priors described above, our goal is to find the *maximum a posteriori* (MAP) estimates $(\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\theta}}) = \arg\max_{(\boldsymbol{\beta},\boldsymbol{\theta})}(p(D|\boldsymbol{\beta},\boldsymbol{\theta})p(\boldsymbol{\beta}|\gamma_1)\, p(\boldsymbol{\theta}|\gamma_2))$, where $p(D|\boldsymbol{\beta},\boldsymbol{\theta})$ is the likelihood of the training data, which has the following form (noting that $1 - \Phi(z) = \Phi(-z)$):

$$p(D|\boldsymbol{\beta},\boldsymbol{\theta}) = \prod_{i=1}^{n} \left[\Phi\left(\beta_0 + \sum_{j=1}^{n} \beta_i K_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})\right)\right]^{y^{(i)}} \left[\Phi\left(-\beta_0 - \sum_{j=1}^{n} \beta_i K_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})\right)\right]^{(1-y^{(i)})}.$$

We next describe an EM algorithm that finds MAP estimates using the hierarchical prior models described above and the latent variable interpretation of the probit model [1], [4]. Consider the $(n+1)$-dimensional vector function $\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}) = [1, K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^{(1)}), \ldots, K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^{(n)})]^T$, and the random function $z(\boldsymbol{x}, \boldsymbol{\beta}, \boldsymbol{\theta}) = \boldsymbol{h}_{\boldsymbol{\theta}}^T(\boldsymbol{x})\boldsymbol{\beta} + w$, where $w \sim \mathcal{N}(0,1)$. If a classifier were to assign the label $y = 1$ to an example $\boldsymbol{x}$ whenever $z(\boldsymbol{x}, \boldsymbol{\beta}, \boldsymbol{\theta}) \geq 0$ and $y = 0$ whenever $z(\boldsymbol{x}, \boldsymbol{\beta}, \boldsymbol{\theta}) < 0$, then we would recover the probit model:

$$P(y = 1|\boldsymbol{x}, \boldsymbol{\beta}, \boldsymbol{\theta}) = P\big(\boldsymbol{h}_{\boldsymbol{\theta}}^T(\boldsymbol{x})\boldsymbol{\beta} + w > 0\big) = \Phi\big(\boldsymbol{h}_{\boldsymbol{\theta}}^T(\boldsymbol{x})\boldsymbol{\beta}\big). \tag{8}$$

Now, consider the vector of missing variables $\boldsymbol{z} = [z^{(1)}, \ldots, z^{(n)}]^T$, where $z^{(i)} = z(\boldsymbol{x}^{(i)}, \boldsymbol{\beta}, \boldsymbol{\theta})$. Consider also the vectors of missing variables $\boldsymbol{\tau} = [\tau_0, \ldots, \tau_n]^T$ and $\boldsymbol{\rho} = [\rho_1, \ldots, \rho_p]^T$. If $\boldsymbol{z}$, $\boldsymbol{\tau}$, and $\boldsymbol{\rho}$ were known, estimating $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ would be much simpler:

- If $\boldsymbol{z}$ was known, we would have the linear observation model $\boldsymbol{z} = \boldsymbol{H}_{\boldsymbol{\theta}}\boldsymbol{\beta} + \boldsymbol{w}$, where $\boldsymbol{H}_{\boldsymbol{\theta}} = [\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}^{(1)}), \ldots, \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}^{(n)})]^T$ is the so-called *design matrix* and $\boldsymbol{w} = [w_1, \ldots, w_n]^T$.
- If $\boldsymbol{\tau}$ and $\boldsymbol{\rho}$ were known, the priors on $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ would be a Gaussian and a truncated Gaussian, respectively, much simpler to handle than a Laplacian and an exponential.

These observations suggest the use of an EM algorithm in which $\boldsymbol{z}$, $\boldsymbol{\tau}$, and $\boldsymbol{\rho}$ play the role of missing variables. The EM algorithm will produce a sequence of estimates $\widehat{\boldsymbol{\beta}}^{(t)}$ and $\widehat{\boldsymbol{\theta}}^{(t)}$ by alternating between the E-step and the M-step, described below.

E-step. The E-step requires computing the expected value of the complete log-posterior (i.e., the one which would be maximized if we had $\boldsymbol{z}$, $\boldsymbol{\tau}$, and $\boldsymbol{\rho}$) conditioned on the data D and the current estimate of the parameters, $\widehat{\boldsymbol{\beta}}^{(t)}$ and $\widehat{\boldsymbol{\theta}}^{(t)}$. This complete log-posterior is given by

$$\begin{aligned} &\log\, p(\boldsymbol{\beta}, \boldsymbol{\theta}|D, \boldsymbol{z}, \boldsymbol{\tau}, \boldsymbol{\rho}) \\ &\quad\propto \log p(\boldsymbol{z}|\boldsymbol{\beta}, \boldsymbol{\theta}) + \log p(\boldsymbol{\beta}|\boldsymbol{\tau}) + \log p(\boldsymbol{\theta}|\boldsymbol{\rho}) &(9) \\ &\quad\propto -\boldsymbol{z}^T\boldsymbol{z} - \boldsymbol{\beta}^T\boldsymbol{H}_{\boldsymbol{\theta}}^T(\boldsymbol{H}_{\boldsymbol{\theta}}\boldsymbol{\beta} - 2\boldsymbol{z}) - \boldsymbol{\beta}^T\boldsymbol{T}\boldsymbol{\beta} - \boldsymbol{\theta}^T\boldsymbol{R}\boldsymbol{\theta}, &(10) \end{aligned}$$

where $\boldsymbol{T}$ and $\boldsymbol{R}$ are diagonal matrices: $\boldsymbol{T} = \operatorname{diag}(\tau_0^{-1}, \ldots, \tau_n^{-1})$ and $\boldsymbol{R} = \operatorname{diag}(\rho_1^{-1}, \ldots, \rho_p^{-1})$. The expected value of this log-posterior, usually denoted as the Q function, is thus

$$Q\Big(\boldsymbol{\beta}, \boldsymbol{\theta}\Big|\widehat{\boldsymbol{\beta}}^{(t)}, \widehat{\boldsymbol{\theta}}^{(t)}\Big) = \mathbb{E}\Big[-\boldsymbol{z}^T\boldsymbol{z} - \boldsymbol{\beta}^T\boldsymbol{H}_{\boldsymbol{\theta}}^T(\boldsymbol{H}_{\boldsymbol{\theta}}\boldsymbol{\beta} - 2\boldsymbol{z}) - \boldsymbol{\beta}^T\boldsymbol{T}\boldsymbol{\beta} - \boldsymbol{\theta}^T\boldsymbol{R}\boldsymbol{\theta}\Big|D, \widehat{\boldsymbol{\beta}}^{(t)}, \widehat{\boldsymbol{\theta}}^{(t)}\Big].$$

where the expectation is, of course, with respect to the missing variables. Since the M-step will maximize Q with respect to $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, the term $-\boldsymbol{z}^T\boldsymbol{z}$ can be dropped, leaving

$$\begin{aligned} Q\Big(\boldsymbol{\beta}, \boldsymbol{\theta}\Big|\widehat{\boldsymbol{\beta}}^{(t)}, \widehat{\boldsymbol{\theta}}^{(t)}\Big) = &- \boldsymbol{\beta}^T\boldsymbol{H}_{\boldsymbol{\theta}}^T\boldsymbol{H}_{\boldsymbol{\theta}}\boldsymbol{\beta} + 2\,\boldsymbol{\beta}^T\boldsymbol{H}_{\boldsymbol{\theta}}^T\mathbb{E}\Big[\boldsymbol{z}\Big|D, \widehat{\boldsymbol{\beta}}^{(t)}, \widehat{\boldsymbol{\theta}}^{(t)}\Big] \\ &- \boldsymbol{\beta}^T\mathbb{E}\Big[\boldsymbol{T}\Big|D, \widehat{\boldsymbol{\beta}}^{(t)}, \widehat{\boldsymbol{\theta}}^{(t)}\Big]\boldsymbol{\beta} - \boldsymbol{\theta}^T\mathbb{E}\Big[\boldsymbol{R}\Big|D, \widehat{\boldsymbol{\beta}}^{(t)}, \widehat{\boldsymbol{\theta}}^{(t)}\Big]\boldsymbol{\theta}. \end{aligned} \tag{11}$$

The E-step thus reduces to computing each of the expectations appearing in (11). For the expectation of $\boldsymbol{z}$, if we define $l^{(i)} = 2y^{(i)} - 1$ (i.e., map the labels to $\{-1, +1\}$), we have

$$v_i \equiv \mathbb{E}\Big[z^{(i)}\Big|D, \widehat{\boldsymbol{\beta}}^{(t)}, \widehat{\boldsymbol{\theta}}^{(t)}\Big] = \boldsymbol{h}_{\boldsymbol{\theta}}^T(\boldsymbol{x}^{(i)})\widehat{\boldsymbol{\beta}}^{(t)} + \frac{l^{(i)}\,\mathcal{N}\Big(\boldsymbol{h}_{\boldsymbol{\theta}}^T(\boldsymbol{x}^{(i)})\widehat{\boldsymbol{\beta}}^{(t)}\Big|0, 1\Big)}{\Phi\Big(l^{(i)}\boldsymbol{h}_{\boldsymbol{\theta}}^T(\boldsymbol{x}^{(i)})\widehat{\boldsymbol{\beta}}^{(t)}\Big)}, \tag{12}$$

which follows from the fact that $z^{(i)}$ is distributed as a Gaussian with mean $\boldsymbol{h}_{\boldsymbol{\theta}}^T(\boldsymbol{x}^{(i)})\widehat{\boldsymbol{\beta}}^{(t)}$, but left-truncated at zero if $y^{(i)} = 1$ and right-truncated at zero if $y^{(i)} = 0$. Some further, simple manipulation allows us to show that the expectations of τ_i^{-1} and ρ_k^{-1} are

$$\omega_i \equiv \mathbb{E}\Big[\tau_i^{-1}\Big|D, \widehat{\boldsymbol{\beta}}_i^{(t)}, \gamma_1\Big] = \frac{\int_0^\infty \tau_i^{-1}p(\tau_i|\gamma_1)\, p\Big(\widehat{\boldsymbol{\beta}}_i^{(t)}\Big|\tau_i\Big)d\tau_i}{\int_0^\infty p(\tau_i|\gamma_1)\, p\Big(\widehat{\boldsymbol{\beta}}_i^{(t)}\Big|\tau_i\Big)d\tau_i} = \gamma_1\Big|\widehat{\boldsymbol{\beta}}_i^{(t)}\Big|^{-1}, \tag{13}$$

$$\delta_k \equiv \mathbb{E}\Big[\rho_k^{-1}\Big|D, \widehat{\boldsymbol{\theta}}_k^{(t)}, \gamma_2\Big] = \gamma_2\Big(\widehat{\boldsymbol{\theta}}_k^{(t)}\Big)^{-1}. \tag{14}$$

In summary, all three expectation terms in the E-step can be computed analytically.

M-step. In this step, the parameter estimates are updated according to

$$\widehat{\boldsymbol{\beta}}^{(t+1)}, \widehat{\boldsymbol{\theta}}^{(t+1)} = \arg\max_{\boldsymbol{\beta},\boldsymbol{\theta}} Q\Big(\boldsymbol{\beta}, \boldsymbol{\theta}\Big|\widehat{\boldsymbol{\beta}}^{(t)}, \widehat{\boldsymbol{\theta}}^{(t)}\Big). \tag{15}$$

After defining $\boldsymbol{v} = [v_1, \ldots, v_n]^T$, $\boldsymbol{\Omega} = \operatorname{diag}(\omega_0, \ldots, \omega_n)$, and $\boldsymbol{\Delta} = \operatorname{diag}(\delta_1, \ldots, \delta_p)$, the M-step requires that we maximize the following function with respect to $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ jointly:

$$Q\Big(\boldsymbol{\beta}, \boldsymbol{\theta}\Big|\widehat{\boldsymbol{\beta}}^{(t)}, \widehat{\boldsymbol{\theta}}^{(t)}\Big) = -\boldsymbol{\beta}^T\boldsymbol{H}_{\boldsymbol{\theta}}^T\boldsymbol{H}_{\boldsymbol{\theta}}\boldsymbol{\beta} + 2\boldsymbol{\beta}^T\boldsymbol{H}_{\boldsymbol{\theta}}^T\boldsymbol{v} - \boldsymbol{\beta}^T\boldsymbol{\Omega}\,\boldsymbol{\beta} - \boldsymbol{\theta}^T\boldsymbol{\Delta}\boldsymbol{\theta}. \tag{16}$$

The gradient with respect to $\boldsymbol{\beta}$, and the partial derivatives with respect to each θ_k are

$$\nabla_{\boldsymbol{\beta}} Q = -2\boldsymbol{H}_{\boldsymbol{\theta}}^T \boldsymbol{H}_{\boldsymbol{\theta}} \boldsymbol{\beta} + 2\boldsymbol{H}_{\boldsymbol{\theta}}^T \boldsymbol{v} - 2\boldsymbol{\Omega}\boldsymbol{\beta} \tag{17}$$

$$\frac{\partial Q}{\partial \theta_k} = -2\delta_k \theta_k - 2\sum_{i=1}^{n}\sum_{j=1}^{n+1}\left\{(\boldsymbol{H}_{\boldsymbol{\theta}}\boldsymbol{\beta} - \boldsymbol{v})\boldsymbol{\beta}^T \odot \left(\frac{\partial \boldsymbol{H}_{\boldsymbol{\theta}}}{\partial \theta_k}\right)\right\}_{(i,j)} \tag{18}$$

where $\odot$ represents element-wise Hadamard matrix multiplication. For the polynomial and RBF kernels used in this paper ((3) and (4)), these derivatives are computed trivially.

In general, the joint maximization of Q with respect to $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ cannot be performed analytically. However, we can exploit the fact that, for any given $\boldsymbol{\theta}$, the corresponding optimal $\boldsymbol{\beta}$, denoted $\widehat{\boldsymbol{\beta}}_{\boldsymbol{\theta}}$, can be evaluated analytically by solving $\nabla_{\boldsymbol{\beta}} Q = 0$, which yields

$$\widehat{\boldsymbol{\beta}}_{\boldsymbol{\theta}} = (\boldsymbol{\Omega} + \boldsymbol{H}_{\boldsymbol{\theta}}^T \boldsymbol{H}_{\boldsymbol{\theta}})^{-1} \boldsymbol{H}_{\boldsymbol{\theta}}^T \boldsymbol{v} = \boldsymbol{S}(\boldsymbol{I} + \boldsymbol{S}\boldsymbol{H}_{\boldsymbol{\theta}}^T \boldsymbol{H}_{\boldsymbol{\theta}} \boldsymbol{S})^{-1} \boldsymbol{S}\boldsymbol{H}_{\boldsymbol{\theta}}^T \boldsymbol{v}, \tag{19}$$

where $\boldsymbol{S} = \mathrm{diag}(s_0, \ldots, s_n)$ with $s_i \equiv \omega_i^{-1/2} = \gamma_1^{-1/2}|\widehat{\boldsymbol{\beta}}_i^{(t)}|^{1/2}$. Matrix $\boldsymbol{S}$ allows a stable numerical implementation since the sparsity-promoting priors will drive several of the β_i to zero, thereby causing numerical problems in implementations that use $\boldsymbol{\Omega}$ directly. Having solved the maximization analytically with respect to $\boldsymbol{\beta}$, we are left with a function only of $\boldsymbol{\theta}$, resulting from plugging $\widehat{\boldsymbol{\beta}}_{\boldsymbol{\theta}}$ into Q. Maximization with respect to $\boldsymbol{\theta}$ can be handled by any standard method; in the results presented below, we use a simple conjugate gradient algorithm.

3.1 Approximate Methods for Improving Computational Complexity

The computational complexity of the algorithm described above is very reasonable for moderately sized problems where the number of training examples n and the feature dimensionality p can each be on the order of several hundred. The computational bottleneck of the algorithm is the matrix inversion in (19), which becomes impractical for large numbers of training examples n; this problem is endemic to kernel methods and reduced set strategies may be useful in these situations [11] (see also Section 5.2).

On the other hand, when we have a few tens of training samples but several thousand feature dimensions (so-called "small n, large p" problems), we can apply an initial feature preselection step using some alternative approach and, subsequently, invoke the JCFO algorithm on the feature subset. For example, this feature preselection can easily be performed using JCFO itself, but in a nonkernelized mode: setting all θ_k equal to some constant and letting $\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}) = [1, x_1, x_2, \ldots, x_p]^T$. In such a nonkernelized mode, our JCFO algorithm reduces to the algorithm of [4], and very efficient implementations that avoid any matrix inversions can be obtained. The sparsity-promoting prior on $\boldsymbol{\beta}$ yields a linear hyperplane classifier using only a few features. These features can then be used in a fully kernelized JCFO. In the experiments reported below, this strategy is employed only with the very high-dimensional gene expression data sets.

4 Experimental Results

Following standard procedure in experiments with kernel classifiers, each data set is initially normalized to have zero mean and unit variance. The regularization parameters γ_1 and γ_2 (controlling the degrees of sparsity enforced on $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, respectively) are selected by cross-validation on the training data. To obtain meaningful comparisons against other published algorithms, we use either identical cross-validation procedures or the same training and test sets as those previously reported. We present results of experiments with both high-dimensional data sets containing many irrelevant features, and low-dimensional data sets in which feature selection is not critical or sometimes even useful. Before doing so, however, we consider a series of tests with synthetic data to illustrate the ability of JCFO to guard against the curse of dimensionality.

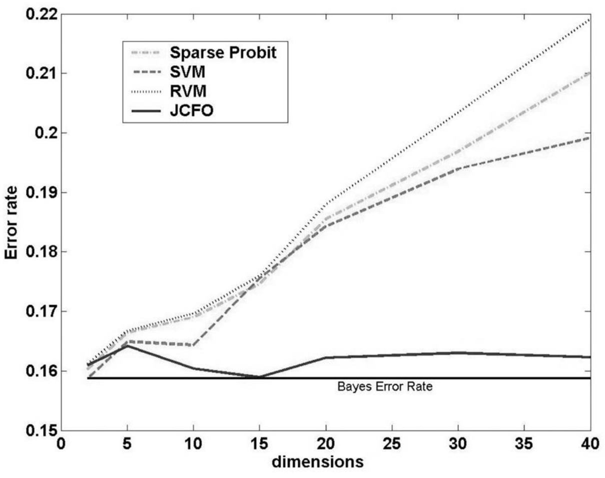


Fig. 1. Effect of increasing numbers of irrelevant features added to a series of synthetic data sets. In accordance with the curse of dimensionality, the error rate of most kernel classifiers increases as more irrelevant features are introduced, but our JCFO algorithm is largely immune.

4.1 Effect of Irrelevant Predictor Variables

The first experiment aims at illustrating the robustness of JCFO against the inclusion of increasing numbers of irrelevant features. To this end, we generate synthetic data from one of two normal distributions with unit variance, depending on the class. The locations in p-dimensional space of the means of these two distributions, $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, are given by

$$\boldsymbol{\mu}_1 = -\boldsymbol{\mu}_2 = [1/\sqrt{2}, 1/\sqrt{2}, \underbrace{0, 0, \ldots, 0}_{(p-2)\text{ zeros}}]^T.$$

Regardless of the feature dimensionality p, the optimal classifier is linear and uses only the first two dimensions of the data, and the Bayes error rate is $\Phi(-1) \simeq 0.1587$. The competing algorithms are given training sets with 100 examples from each class and tested on independent test sets with 500 examples from each class. The results are averaged over 20 random samples of the training set and the entire procedure is repeated for several feature dimensions p. The average error rates of the competing algorithms, all with linear kernels, are compared in Fig. 1. Although error bars have been omitted from Fig. 1 to enhance readability, for $p \geq 10$, JCFO is assessed by a binomial test to be statistically superior to all other competing algorithms with a p-value of 0.05 or better.

This experiment shows that JCFO is much more robust to the presence of increasing numbers of irrelevant features than other kernel methods, such as the SVM, RVM, and SPR. Among the kernel methods compared in this experiment, only JCFO is able to estimate the scale of the input features, so it is not surprising that it is the only one largely immune to the presence of irrelevant variables. Furthermore, the feature

TABLE 1
Accuracy of Diagnostic Classification with Gene Expression Data Sets (% Correct)

Classifier	AML/ALL	Colon
Adaboosting (Decision stumps) [2]	95.8	72.6
SVM (Linear kernel) [2]	94.4	77.4
SVM (Quadratic kernel) [2]	95.8	74.2
Logistic regression (No kernel: on feature space) [9]	97.2	71.0
RVM (No kernel: on feature space) [9]	97.2	88.7
Sparse probit regression (No kernel: on feature space)	97.2	88.7
Sparse probit regression (Linear kernel)	97.2	91.9
Sparse probit regression (Quadratic kernel)	95.8	84.6
JCFO (Linear kernel)	100.0	96.8
JCFO (Quadratic kernel)	98.6	88.7

selection ability of our JCFO algorithm was clearly demonstrated in this experiment: For almost all the random draws of the training set across all values of p, JCFO identified only the two relevant features by choosing $\theta_k = 0$ for all $k > 2$. The rapid performance degradation of the other kernel methods shows that feature selection is indeed crucial, even for large margin techniques like the SVM.

A similar experiment was also performed in [4] using *nonkernelized* sparse probit regression in which $\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}) = [1, x_1, \dots, x_p]^T$. The results reported therein demonstrate reduced performance degradation with increasing feature dimensionality, lending support to the two-stage approximate method described in Section 3.1. As shown in the next section, in the context of data with an extremely high feature dimensionality, the two-stage approximate version of JCFO outperforms nonkernelized sparse probit regression alone.

4.2 Results with High-Dimensional Gene Expression Data Sets

Gene expression data sets are known to contain a large number of irrelevant and redundant variables; consequently, effective feature selection becomes a key factor in determining the performance of a classification algorithm. In fact, the strategy for learning a classifier is likely to be less relevant here than the choice of feature selection method. We consider two commonly-analyzed data sets. The first contains expression levels of 7,129 genes from 47 patients with acute myeloid leukemia (AML) and 25 patients with acute lymphoblastic leukemia (ALL).[2] The second contains expression levels of 2,000 genes from 40 tumor and 22 normal colon tissues.[3] Due to the high feature dimensionality of these data sets, we used low-order polynomial kernels in this set of experiments.

To assess diagnostic accuracy, we use a full leave-one-out cross-validation (LOOCV) procedure: We train on $n - 1$ samples and test the obtained classifier on the remaining sample, and repeat this procedure for every sample. To reduce the computational cost of the full LOOCV procedure, we accelerate JCFO by using the approximate method outlined in Section 3.1. This allows performing full LOOCV error rate estimation on an 800MHz Pentium III machine in a couple of hours. The results are reported in Table 1, where JCFO is compared against Adaboost, SVM, RVM, SPR, and logistic regression. On the AML/ALL data set, although JCFO outperforms all the other methods, the difference between it and the other methods is too small to be statistically significant given the small sample size. However, on the Colon data set, a binomial test indicates that JCFO is superior to kernelized SPR with a p-value of 0.12 and superior to all the other methods with a p-value of 0.03 or better.

Since JCFO is also able to identify the most relevant features, we can assess which genes were identified as particularly useful for good classification performance. Only about 20 genes were identified as diagnostically important by JCFO ($\theta_k > 0$); importantly, the role of many of these genes has already been described in the medical literature (as discussed in [10]).

4.3 Results with Low-Dimensional Benchmark Data Sets

In the second set of experiments with real data, we assess the performance of JCFO using a few popular benchmark data sets for nonlinear classifier design. It is worth emphasizing that these data sets are of relatively low feature dimensionality and were not deliberately created to contain irrelevant predictor variables. Detailed descriptions of these data sets, as well as availability information, can be found in [4]. Table 2 reports the accuracy of JCFO for several benchmark data sets alongside some of the best results reported in the literature. Since Gaussian RBF kernels are used in all these kernel classifier results from the literature, we use only Gaussian RBF kernels here; we also show results reported in the literature for Bayesian training of linear discriminants, neural networks, and logistic regression.

For the RVM results from [18], the kernel width parameter (equivalent to setting $\boldsymbol{\theta}$ to some constant) is chosen by cross-validation. For SPR, the kernel widths are set to the same values reported in [4] (these kernel widths

2. The AML/ALL data set is available from http://www-genome.wi.mit.edu/mpr/table_AML_ALL_samples.rtf.

3. The Colon data set is available from http://microarray.princeton.edu/oncology/affydata/index.html.

TABLE 2
Accuracy of Diagnostic Classification with Benchmark Data Sets (Average Number of Errors)

Classifier	Ripley	Pima	Crabs	WBC
Linear discriminant [14]	N/A	67	3	19
Neural network [20]	N/A	75	3	N/A
Gaussian process [20]	92	67	3	8
SVM [14]	106	64	4	9
Logistic regression [20]	N/A	66	4	N/A
RVM [18]	93	65	0	9
Sparse probit regression	95	62	0	9
JCFO	92	64	0	8

were observed to be optimal for SPR in the sense of minimizing test error rates); JCFO training procedures are initialized with these same widths. For the SVM results from [14], as well as for the Gaussian process results [20], the scales of the individual feature dimensions are estimated within the learning algorithm.

Our algorithm did not perform any feature selection in the Ripley data set, selected five out of eight variables in the Pima data set, and selected three out of five variables in the Crabs data set. These results are similar to those reported in [14]. JCFO is also very sparse in its utilization of kernel functions: It chooses an average of 4 out of 100, 5 out of 200, 5 out of 80, and 6 out of 300 kernels for the Ripley, Pima, Crabs, and WBC data sets, respectively. In short, it successfully finds classifiers that are sparse both in $\boldsymbol{\beta}$ and in $\boldsymbol{\theta}$.

Our results are either the best or very close to the best in each case, but the differences are not statistically significant. However, although all the state-of-the-art methods considered here are competitive in terms of error rates on these data sets, JCFO is able to successfully perform feature selection at the same time. Two of the competing methods (the SVM of [14] and the Gaussian process of [20]) also perform feature scale estimation. Since the feature selection property was the principal difference between JCFO and these two ARD algorithms, their similar performance confirms that JCFO does at least as well, even when applied to data sets that are not particularly amenable to feature selection.

5 Conclusion

5.1 Discussion

JCFO should be judged on two different criteria: accurate classification and parsimonious feature selection. Like some ARD methods, JCFO estimates feature scaling coefficients along with the other parameters of the classifier; it differs from those algorithms in its use of sparsity-promoting priors that encourage many of the θ_k to be exactly zero. Our experimental results indicate that, for high-dimensional data with many irrelevant features, the classification accuracy of JCFO is likely to be statistically superior to other methods. Even for low-dimensional data where we do not expect feature selection to be critical or sometimes even useful, JCFO seems to be at least as accurate as other state-of-the-art methods that also perform feature scale estimation along with classifier design. However, in addition to high accuracy, JCFO has the additional benefit of more parsimonious feature selection. This is especially important in the context of gene expression data, where many of the features selected by JCFO have been previously identified as clinically relevant for diagnosis.

5.2 Future Work

The joint optimization of the classifier parameters and the feature scale parameters comes at a price: The JCFO algorithm is significantly slower than the SPR algorithm of [4], although its running time is still acceptable for the data sets studied in this paper. However, the computational complexity of JCFO might render it impractical if naïvely applied to data sets with several hundred features or several thousand samples. This computational issue is endemic in almost all kernel-based methods; while the approximate method described in Section 3.1 partially alleviates the problem with high-dimensional feature spaces, it is not helpful in problems with large training sets. We are currently working on alternative techniques to improve the computational efficiency of the algorithm. In one promising avenue of research, we are investigating the use of the *fast Gauss transform* [15] and reduced set methods [11] to dramatically improve the computational cost of (19).

In conclusion, JCFO has successfully achieved both of its objectives: excellent classification accuracy and automatic sparse feature selection. Nevertheless, we must still address some important questions. Does our EM formulation converge to a global maximum or do we face multiple local maxima? Can we obtain tight theoretical upper bounds on the error rate that can be used to motivate further improvements in algorithm? We are currently investigating different ways to address these issues. Code developed as part of this research can be freely obtained from the first author for academic research use.

Acknowledgments

The authors gratefully acknowledge the anonymous reviewers for very detailed and helpful suggestions for improving the original manuscript.

References

[1] J. Albert and S. Chib, “Bayesian Analysis of Binary and Polychotomous Response Data,” *J. Am. Statistical Assoc.*, vol. 88, pp. 669-679, 1993.
[2] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini, “Tissue Classification with Gene Expression Profiles,” *Proc. Fourth Ann. Int'l Conf. Computational Molecular Biology (RECOMB 2000),* 2000.
[3] R. Duda, P. Hart, and D. Stork, *Pattern Classification.* New York: John Wiley & Sons, 2001.
[4] M.A.T. Figueiredo, “Adaptive Sparseness for Supervised Learning,” *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, pp. 1150-1159, 2003.
[5] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, “Gene Selection for Cancer Classification Using Support Vector Machines,” *Machine Learning,* vol. 46, nos. 1-3, pp. 389-422, 2002.
[6] R. Herbrich, *Learning Kernel Classifiers.* Cambridge, Mass.: MIT Press, 2002.
[7] D. Husmeier, W. Penny, and S.J. Roberts, “An Empirical Evaluation of Bayesian Sampling with Hybrid Monte Carlo for Training Neural Network Classifiers,” *Neural Networks,* vol. 12, pp. 677-705, 1999.

[8] D. Husmeier and S.J. Roberts, "Regularisation of RBF-Networks with the Bayesian Evidence Scheme," *Proc. Int'l Conf. Artificial Neural Networks (ICANN99)*, pp. 533-538, 1999.

[9] B. Krishnapuram, A.J. Hartemink, and L. Carin, "Logistic Regression and RVM for Cancer Diagnosis from Gene Expression Signatures," *Proc. 2002 Workshop Genomic Signal Processing and Statistics (GENSIPS)*, 2002.

[10] B. Krishnapuram, L. Carin, and A.J. Hartemink, "Joint Classifier and Feature Optimization for Cancer Diagnosis Using Gene Expression Data," *Proc. Seventh Ann. Int'l Conf. Computational Molecular Biology (RECOMB 2003)*, 2003.

[11] Y.-J. Lee and O.L. Mangasarian, "RSVM: Reduced Support Vector Machines," *Proc. SIAM Int'l Conf. Data Mining*, 2001.

[12] P. McCullagh and J. Nelder, *Generalized Linear Models.* London: Chapman and Hall, 1989.

[13] R.M. Neal, *Bayesian Learning for Neural Networks.* New York: Springer Verlag, 1996.

[14] M. Seeger, "Bayesian Model Selection for Support Vector Machines, Gaussian Processes, and Other Kernel Classifiers," *Proc. Advances in Neural Information Processing Systems (NIPS) 12*, 2000.

[15] X. Sun and Y. Bao, "A Kronecker Product Representation of the Fast Gauss Transform," *SIAM J. Matrix Analysis and Applications*, vol. 24, no. 3, pp. 768-786, 2003.

[16] B. Schölkopf and A. Smola, *Learning with Kernels.* Cambridge, Mass.: MIT Press, 2002.

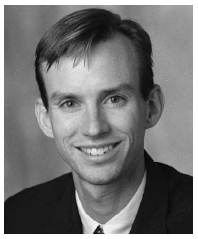
[17] R. Tibshirani, "Regression Shrinkage and Selection via the LASSO," *J. Royal Statistical Soc. (B)*, vol. 58, pp. 267-288, 1996.

[18] M.E. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *J. Machine Learning Research*, vol. 1, pp. 211-244, 2001.

[19] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature Selection for SVMs," *Proc. Advances in Neural Information Processing Systems (NIPS) 12*, 2000.

[20] C.K.I. Williams and D. Barber, "Bayesian Classification with Gaussian Processes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1342-1351, Dec. 1998.

Balaji Krishnapuram received the BTech degree from the Indian Institute of Technology (IIT), Kharagpur, in 1999 and the MS degree from Duke University in 2001, both in electrical engineering. He is currently pursing the PhD degree in electrical engineering at Duke University. His current research interests include statistical pattern recognition and computational learning theory. He is also interested in applications in automatic target detection, signal processing, computer vision, and bioinformatics.



Alexander J. Hartemink received the SM degree in 1997 and the PhD degree in 2001 from the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology (MIT) as a US National Science Foundation Graduate Research Fellow. Before that, he earned the MPhil degree in economics in 1996 from Oxford University as a Rhodes Scholar and the BS degree in mathematics, the BS degree in physics, and the AB degree in economics in 1994 from Duke University as an Angier B. Duke Scholar. He is currently an assistant professor of computer science at Duke, and is affiliated with the Center for Bioinformatics and Computational Biology, one of five centers in Duke's Institute for Genome Sciences and Policy. His research centers on the development and application of principled computational and statistical methods for understanding complex biological systems, with a particular focus on the use of techniques from statistical and machine learning to address problems in functional genomics and systems neurobiology.

Lawrence Carin (F'01, SM'96) received the BS, MS, and PhD degrees in electrical engineering from the University of Maryland, College Park, in 1985, 1986, and 1989, respectively. In 1989, he joined the Electrical Engineering Department at Polytechnic University (Brooklyn) as an assistant professor and became an associate professor in 1994. In September 1995, he joined the Electrical Engineering Department at Duke University, where he is now the William H. Younger Professor of Engineering. Dr. Carin was the principal investigator (PI) on a Multidisciplinary University Research Initiative (MURI) on demining (1996-2001) and he is currently the PI of a MURI dedicated to multimodal inversion. He is an associate editor of the *IEEE Transactions on Antennas and Propagation*. His current research interests include short-pulse scattering, subsurface sensing, and wave-based signal processing. He is a member of the Tau Beta Pi and Eta Kappa Nu honor societies. He is a fellow of the IEEE.

Mário A.T. Figueiredo (S'87, M'95, SM'00) received the EE, MSc, and PhD degrees in electrical and computer engineering, all from the Instituto Superior Tecnico (IST), the engineering school of the Technical University of Lisbon, Portugal, in 1985, 1990, and 1994, respectively. Since 1994, he has been with Department of Electrical and Computer Engineering, IST. He is also a researcher and area coordinator at the Institute of Telecommunications, Lisbon. In 1998, he held a visiting position with the Department of Computer Science and Engineering at Michigan State University, East Lansing. His scientific interests include image processing and analysis, computer vision, statistical pattern recognition, and statistical learning. He received the Portuguese IBM Scientific Prize in 1995 for work on unsupervised image restoration. He is an associate editor of the journals *Pattern Recognition Letters*, *IEEE Transactions on Image Processing*, and *IEEE Transactions on Mobile Computing*. He is also guest coeditor of special issues of the journals *IEEE Transactions on Pattern Analysis and Machine Intelligence* and *IEEE Transactions on Signal Processing*. He was a cochair of the 2001 and 2003 Workshops on Energy Minimization Methods in Computer Vision and Pattern Recognition and is the local chair of the 2004 Joint IAPR International Workshops on Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition. He has been a member of the program committees of several international conferences, including CVPR, EECV, ICASSP, and ICPR. He is a senior member of the IEEE.