Joint Classifier and Feature Optimization for Cancer Diagnosis Using Gene Expression Data

Balaji Krishnapuram
Dept. of Elec. Engineering
Duke University
Durham, NC 27708
balaji@ee.duke.edu

Lawrence Carin
Dept. of Elec. Engineering
Duke University
Durham, NC 27708
lcarin@ee.duke.edu

Alexander J. Hartemink
Dept. of Computer Science
Duke University
Durham, NC 27708
amink@cs.duke.edu

ABSTRACT

Recent research has demonstrated quite convincingly that accurate cancer diagnosis can be achieved by constructing classifiers that are designed to compare the gene expression profile of a tissue of unknown cancer status to a database of stored expression profiles from tissues of known cancer status. This paper introduces the JCFO, a novel algorithm that uses a sparse Bayesian approach to jointly identify both the optimal nonlinear classifier for diagnosis and the optimal set of genes on which to base that diagnosis. We show that the diagnostic classification accuracy of the proposed algorithm is superior to a number of current state-of-the-art methods in a full leave-one-out cross-validation study of two widely used benchmark datasets. In addition to its superior classification accuracy, the algorithm is designed to automatically identify a small subset of genes (typically around twenty in our experiments) that are capable of providing complete discriminatory information for diagnosis. Focusing attention on a small subset of genes is not only useful because it produces a classifier with good generalization capacity, but also because this set of genes may provide insights into the mechanisms responsible for the disease itself. A number of the genes identified by the JCFO in our experiments are already in use as clinical markers for cancer diagnosis; some of the remaining genes may be excellent candidates for further clinical investigation. If it is possible to identify a small set of genes that is indeed capable of providing complete discrimination, inexpensive diagnostic assays might be widely deployable in clinical settings.

Categories and Subject Descriptors

J.3 [**Computer Applications**]: Life and Medical Sciences; I.5.1 [**Computing Methodologies**]: Pattern Recognition—*Models*; I.5.2 [**Computing Methodologies**]: Pattern Recognition—*Design Methodology*

RECOMB'03, April 10–13, 2003, Berlin, Germany.

General Terms

Algorithms; Design; Experimentation

Keywords

disease diagnosis, classification, feature selection, joint optimization, sparse Bayesian methods, JCFO, RVM, SVM

1. INTRODUCTION

In an effort to improve the accuracy of cancer diagnosis and enable the prediction of patient response to different treatment options, a considerable amount of research effort has recently been expended to develop methods that are capable of leveraging the availability of databases of gene expression profiles collected from various classes of cancers [9]. While a large number of supervised and unsupervised methods from the pattern recognition literature have been proposed in this context, techniques based on linear support vector machines (SVM) have proven to be the most popular, and also quite accurate [3, 8].

Other recent research has shown that the expression levels of fewer than ten genes are often sufficient for accurate diagnosis of most cancers, even though the expression levels of a large number of genes are strongly correlated with the disease [7, 14]. In fact, the use of a much larger set of gene expression levels has been shown to have a deleterious effect on the diagnostic accuracy due to the phenomenon known as the *curse of dimensionality.* By identifying a small subset of genes on which to base a diagnosis, we can not only achieve improved diagnostic accuracy, but also gain possibly significant insights into the nature of the disease and the genetic mechanisms responsible for it. In addition, assays that require very few gene expression levels to be measured in order to make a diagnosis are far more likely to be widely deployed in a clinical setting.

In this paper, we develop a Bayesian generalization of the SVM that jointly and simultaneously identifies the optimal nonlinear classifier and selects the optimal set of features (in this case, the genes involved in the diagnosis) via the optimization of a single Bayesian likelihood function. This Joint Classifier and Feature Optimization (JCFO) algorithm implements feature selection by first associating a positive scaling factor with the expression level of each gene. Then during the training phase, along with the identification of the optimal classifier, the JCFO jointly estimates the optimal scaling factors with a strong prior preference for setting most of them to zero. Since setting the scaling factor for

a gene to zero is equivalent to removing its effect on the classifier, the algorithm typically picks out only a handful of genes (typically around twenty in our experiments) that are actually used in the diagnosis. At an abstract and intuitive level, the final classifier itself is simply a form of weighted voting based on the similarity of the gene expression profile of an unlabeled sample to the gene expression profiles of prototypical class examples that are identified during the classifier design. In the JCFO, the similarity between profiles is measured using only the expression levels of the small subset of genes with non-zero scaling factors.

Our method differs in a couple of ways from earlier approaches that have been taken to the problem of identifying the genes that provide maximum diagnostic capability. Specifically, previous work has for the most part focused on the problem of feature selection in isolation from the problem of classifier design; typically features (genes) are first selected, and then all those features are used to design a classifier for producing the diagnosis. By combining these two problems and solving them together as part of a *joint* optimization, we seek to satisfy the most fundamental requirement of feature selection, namely that we retain those features that are most useful in performing the classification itself. A further difference between our approach and previous work is that we address the feature selection problem by using sparsity-promoting priors as an integral part of the objective function used during the training procedure for designing the classifier. This obviates the need to first select the genes that provide maximal diagnostic accuracy on the basis of a full leave-one-out cross-validation (LOOCV) study, and then evaluate the accuracy of the resulting classifiers by performing another full LOOCV on the same data.

The JCFO is able to achieve improved generalization performance because it uses Bayesian priors to promote sparseness in both the selection of genes, as discussed above, and the basis functions used in the classifier. Several algorithms proposed in the last few years—including the relevance vector machine (RVM) of Tipping [13]—have used an automatic relevance determination (ARD) technique [12] for selecting either the features or the basis functions of the classifier during its training. More recently, Figuiredo and Jain have proposed an expectation maximization (EM) algorithm for sparse probit regression that achieves a similar result by maximizing a Bayesian *a posteriori* distribution [6]. While this algorithm does very well on a variety of pattern recognition benchmark problems, it only identifies an optimal classifier given a particular set of features. The JCFO we propose here extends the EM algorithm of Figuiredo and Jain by optimizing a Bayesian posterior to simultaneously obtain both the best classifier and the best feature scaling. Due to a number of similarities between the JCFO and the algorithm of Figuiredo and Jain, we have deliberately tried where possible to preserve their notation in the interest of clarifying the presentation for the reader who may already be familiar with their work.

The remainder of this paper is structured as follows. In Section 2, we formalize the basic problem of pattern recognition in cancer diagnosis, introduce the notation used in the remainder of the paper, and discuss the broader context of the proposed approach, as well as its limitations. We introduce the novel JCFO algorithm in Section 3, and compare the performance of both the JCFO and the sparse probit regression algorithm of Figuiredo and Jain to current state-of-the-art classifiers (including the SVM and the RVM) on two widely used cancer diagnosis benchmark datasets in Section 4. We conclude with a discussion of these results in Section 5.

2. PROBLEM FORMULATION

In the traditional pattern recognition literature, the problem of cancer diagnosis using the gene expression profile of a new tissue sample and a database of known gene expression profiles and their diagnoses falls under the general class of *supervised pattern recognition*. Given a database of training samples from N tissues, we have a set of N gene expression profiles $\boldsymbol{x}^{(i)}$ indexed by $i \in \{1, 2, \ldots, N\}$. Each expression profile $\boldsymbol{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}] \in \mathbb{R}^d$ is a d-dimensional vector representing the measured expression levels of of d genes in the tissue sample. The class membership of each database sample is known and is denoted by $y^{(i)}$. In a two-class case (e.g., the tissues are either cancerous or non-cancerous), we can assume without loss of generality that $y^{(i)} \in \{0, 1\}$. Thus, the training set D consists of N sets of expression profiles and their corresponding class membership labels:

$$D = \left\{ \langle \boldsymbol{x}^{(i)}, y^{(i)} \rangle : \boldsymbol{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \{0, 1\} \right\}_{i=1}^{N} \tag{1}$$

Assuming a parametric form for the functional relationship between $\boldsymbol{x}$ and y as $y = f_{\boldsymbol{\alpha}}(\boldsymbol{x})$, during the training phase we seek to find the optimal parameters $\boldsymbol{\alpha}$ based on the evidence of the training data, D. In other words, in this formulation we seek to learn a binary function $f_{\boldsymbol{\alpha}}(\cdot) : \mathbb{R}^d \rightarrow \{0, 1\}$. This is the formulation adopted by the popular SVM classifier. However, it is often desirable not simply to classify $\boldsymbol{x}$ into one of two classes, but to know the degree of confidence for that classification. In such a case, we would be interested in learning a function $g_{\boldsymbol{\alpha}}(\boldsymbol{x})$ taking values in the interval [0,1] (rather than just the set $\{0, 1\}$), which can be interpreted as the probability that x belongs to class 1. For example, in logistic regression:

$$P(y = 1|\boldsymbol{x}) = g_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sigma\left(\alpha_0 + \sum_{i=1}^{N} \alpha_i x_i\right) \tag{2}$$

where $\sigma(z) = \{1 + \exp(-z)\}^{-1}$ is the logistic link function. The advantage of a classifier with a link function that gives class probabilities over a hard classifier like the SVM is that it can be used to obtain different optimal classifiers under different (possibly asymmetric) cost functions. In the case where the cost function is simply the misclassification error, a classifier can be obtained by thresholding $g_{\boldsymbol{\alpha}}(\boldsymbol{x})$ at $\frac{1}{2}$.

In this paper, we consider classification functions of the form:

$$P(y = 1|\boldsymbol{x}) = g_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \Phi\left(\boldsymbol{\beta}^T \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \tag{3}$$

where $\boldsymbol{\alpha} = [\boldsymbol{\beta}^T, \boldsymbol{\theta}^T]^T$ are the parameters, $\Phi(z)$ is the standard Gaussian cumulative distribution function (otherwise known as the probit link function):

$$\Phi(z) = \int_{-\infty}^{z} N(x|0, 1)dx = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp\left(\frac{-x^2}{2}\right) dx \tag{4}$$

and $\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the vector whose scalar elements are the values of the basis functions for the classifier evaluated at $\boldsymbol{x}$ (we assume that the basis functions are parameterized by $\boldsymbol{\theta}$).

To illustrate the physical meaning of $\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})$ more clearly, we consider three special cases of the general formulation that we have just outlined.

Linear classifiers:

$$\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}) = [1, x_1, \ldots, x_d]^T \tag{5}$$

For linear classifiers, the parameterization of the basis functions by $\boldsymbol{\theta}$ is irrelevant. Note that in the case of symmetric misclassification costs, i.e., if we decide that $y = 1$ when $P(y = 1|\boldsymbol{x}) = g_{\boldsymbol{\alpha}}(\boldsymbol{x}) > \frac{1}{2}$, then $\boldsymbol{\beta}^T \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})$ represents the distance of the gene expression profile $\boldsymbol{x}$ from the classifying linear hyperplane. Here the dimensionality of $\boldsymbol{\beta}$ is $M = d+1$.

Nonlinear classifiers:

$$\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}) = [1, \psi_1(\boldsymbol{x}, \boldsymbol{\theta}), \ldots, \psi_k(\boldsymbol{x}, \boldsymbol{\theta})]^T \tag{6}$$

where $\psi_i(\cdot)$ are k nonlinear basis functions of the classifier. Here the dimensionality of $\boldsymbol{\beta}$ is $M = k + 1$ and for symmetric misclassification costs, $\boldsymbol{\beta}^T \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})$ represents the distance of the gene expression profile $\boldsymbol{x}$ from the classifying linear hyperplane in the ψ-space. In other words the nonlinear classification problem in the original feature space (i.e., $\boldsymbol{x}$-space) is transformed into a linear classification problem into another space using the nonlinear vector mapping function between the two spaces $\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})$.

Kernel classifiers:

$$\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \left[1, K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^{(1)}), \ldots, K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^{(N)})\right]^T \tag{7}$$

where $K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^{(i)})$ is some symmetric kernel function parameterized by $\boldsymbol{\theta}$ [5]. Figure 1 depicts the kernel mapping between the feature space containing $\boldsymbol{x}$ and the kernel space containing $\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})$. The kernel basis function $K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^{(i)})$ provides a nonlinear measure of similarity between the gene expression levels of a new unlabeled sample $\boldsymbol{x}$ and a labeled sample from our training database $\boldsymbol{x}^{(i)}$. Here the dimensionality of $\boldsymbol{\beta}$ is $M = N + 1$. This is used in the SVM, where the final classification function is of the form $f(\boldsymbol{x}) = \beta_0 + \sum \beta_i K_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}^{(i)}) = \boldsymbol{\beta}^T \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})$.

In this paper, we will use the parameters $\boldsymbol{\theta}$ to represent the scaling factors associated with the genes. Thus, the dimensionality of $\boldsymbol{\theta}$ is d and we can write $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_d]^T \in \mathbb{R}^d$. Specifically, if $\theta_j = 0$ then our diagnostic classifier does not use any information about the expression level of the j-th gene in the process of making its decision.

While the SVM requires special restrictions to be placed on the kernel function for the kernel to be admissible in the training procedure— $K_{\boldsymbol{\theta}}(\cdot, \cdot)$ has to be a Mercer kernel—the JCFO does not have any such requirements, and the same is true of the RVM and sparse probit regression algorithms as well. In the research presented in this paper, we have used n-th order polynomial kernels in our experiments:

$$K_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \left(1 + \sum_{l=1}^{d} \theta_l x_l^{(i)} x_l^{(j)}\right)^n \tag{8}$$

In this paper, we seek to find classifiers (i.e., to find values of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$) that not only diagnose the presence or absence of cancer accurately, but also do so with very few non-zero elements in either $\boldsymbol{\theta}$ or $\boldsymbol{\beta}$. Sparsity in $\boldsymbol{\theta}$ implies that the classifier implicitly performs feature selection (i.e., it identifies the genes important for the diagnosis), while sparsity in $\boldsymbol{\beta}$ implies that it finds a small subset of prototypical samples that are highly representative of the different classes we seek to distinguish.

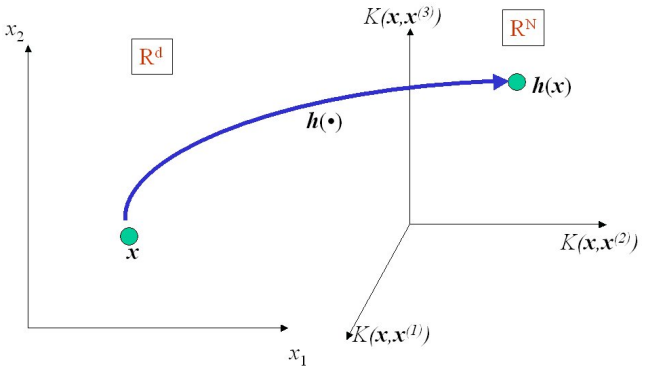

Figure 1: Kernel mapping: a vector x in the feature space of d dimensions is mapped by $\boldsymbol{h}_{\boldsymbol{\theta}}(\cdot)$ into a vector $\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})$ in the kernel space of N dimensions, which is spanned by the N kernel basis functions.

2.1 Bayesian pattern recognition

Our solution can be summarized as follows: the problems of classifier design and feature selection can be viewed together as the single problem of estimating the best parameters $\boldsymbol{\theta}$, $\boldsymbol{\beta}$ from limited data. In order to obtain good *generalization* (i.e., to perform well on new data not seen during training), we need to control the complexity of the learned classifier function. Specifically, we need to guard against two potential problems: if the classifier is too complex it may "learn" irrelevant properties of the particular dataset on which it is trained, and not perform well on as yet unseen data (*overfitting*). On the other hand, if the classifier is too simple, then we may be unable to effectively capture the essential structure of the underlying relationship (*underfitting*). In a Bayesian approach, we solve this problem by introducing some kind of prior knowledge into the design phase.

More precisely, we accomplish this by choosing prior probability distributions over the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ to reflect our (subjective) beliefs about them before seeing any data. In our case, we choose priors that reflect our belief that both $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ are sparse, i.e., $P(\boldsymbol{\theta}, \boldsymbol{\beta})$ is large when most of these parameters are exactly zero (as opposed to being *nearly* zero). This suggests that the prior distributions must drop off very fast as the parameters $\boldsymbol{\theta}$, $\boldsymbol{\beta}$ move away from zero, making density functions that are smoothly differentiable at zero—like the Gaussian—inappropriate.

After seeing the data D, we can use Bayes rule to obtain a posterior distribution $P(\boldsymbol{\theta}, \boldsymbol{\beta}|D)$. The posterior will reflect our final opinion about $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$, taking into account both our prior subjective knowledge (sparsity of the solution) and the evidence provided by the data. Thus, the optimal classifier can be identified by finding the maximum *a posteriori* (MAP) estimate of $\boldsymbol{\theta}$, $\boldsymbol{\beta}$.

The problem, however, is that finding the MAP value of the posterior for our parameters can be a computationally expensive task if we choose arbitrary sparsity-promoting priors. In general, we would be forced to adopt expensive MCMC techniques. Our algorithm builds upon the prior work of Figuiredo and Jain [6] and Tipping [13], both of

whom have developed methods to find a sparse estimate of $\boldsymbol{\beta}$ under certain priors. In particular, we extend the algorithm of Figuiredo and Jain to jointly design the classifier and the feature scaling, i.e., to estimate $\boldsymbol{\theta}$, $\boldsymbol{\beta}$.

In this paper, we choose a Laplacian prior that allows us to design an elegant EM algorithm to find a maximum of the posterior probability density. Using an EM algorithm to optimize the posterior probability immediately raises the concern that any maximum we find may be simply a local maximum rather than global maximum. To investigate this, we have sampled the Hessian of the objective function at a large number of points to test for positive definiteness, and based on this sampling, believe that our objective function has only a single global maximum, at least for polynomial kernels. However, we do not yet have an analytical proof of this, and sampling evidence can never be conclusive since it is possible that we may have inadvertantly sampled in a limited region in which the Hessian is positive definite. Regardless of whether the posterior maximum we reach is local or global, our experiments on widely used benchmark datasets indicate that the JCFO has better diagnostic classification accuracy than other current state-of-the-art methods reported in the literature.

Finally, it is worth noting that sparseness in $\boldsymbol{\beta}$ for kernel classifiers is known to be an important indicator of the *capacity* of the classifier, which measures its generalization. As is evident from the curse of dimensionality (i.e., the risk of overfitting increases as the dimensionality of data increases relative to the number of training samples), sparseness in the features as governed by $\boldsymbol{\theta}$ is also an important factor in increasing the robustness of the classifier design. Thus, our choice of sparsity-promoting priors reflects our desire to obtain a robust classifier that performs well on as-yet-unseen test data.

3. JOINT CLASSIFIER AND FEATURE OPTIMIZATION

As the first step of our Bayesian analysis, we have to specify the prior on the parameters $\boldsymbol{\theta}$, $\boldsymbol{\beta}$ that we want to estimate. We choose to adopt a Laplacian prior on $\boldsymbol{\beta}$, since it is known from earlier work that this prior promotes sparseness (making several $\beta_i = 0$) due to its use of the L_1 norm penalty:

$$P(\boldsymbol{\beta}|\eta) = \prod_{i=1}^{M} \frac{\eta}{2} \exp(-\eta\,|\beta_i|) = \left(\frac{\eta}{2}\right)^M \exp(-\eta\,\|\boldsymbol{\beta}\|_1) \quad (9)$$

Figure 2 illustrates this property of a Laplacian prior, and contrasts it with a Gaussian prior, whose derivative at zero is zero. As the figure illustrates, the difference between $P(0)$ and $P(\beta_i)$ for small β_i is much larger for a Laplacian than for a Gaussian. As a result, if we use a Laplacian prior, learning procedures that seek to maximize the posterior would explicitly favor values of β_i that are exactly 0 instead of small values close to 0, thus promoting sparseness in $\boldsymbol{\beta}$.

If we attempt to use a Laplacian prior directly, we run into computational difficulties. However, equivalently we may instead use the two-level hierarchical model described below, which corresponds to an effective Laplacian prior. Let us consider a model where each β_i is given a zero-mean Gaussian prior with its own variance τ_i:

$$P(\beta_i|\tau_i) = N(\beta_i|0, \tau_i) \quad (10)$$

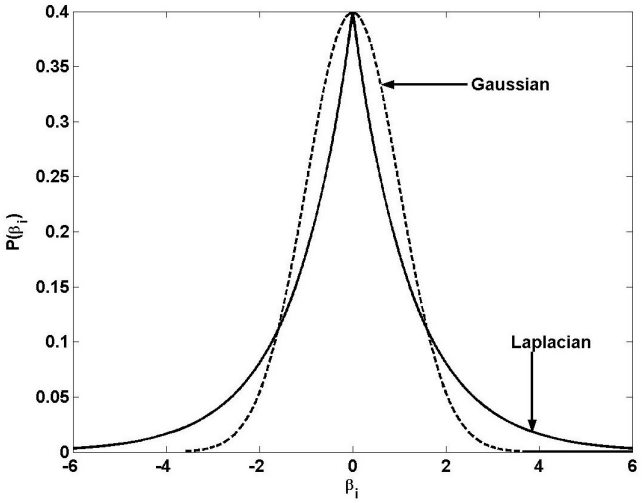

Figure 2: Sparsity-promoting Laplacian prior: the Laplacian distribution is sharply peaked and not smoothly differentiable about zero, unlike the Gaussian distribution, which is everywhere smoothly differentiable and whose derivative at zero is zero.

Further, suppose the variances τ_i have an exponential distribution as their hyperprior:

$$P(\tau_i|\gamma_1) = \frac{\gamma_1}{2} \exp\left(-\frac{\gamma_1 \tau_i}{2}\right), \text{ for } \tau_i \geqslant 0 \quad (11)$$

Thus, the effective prior can be obtained by integrating out τ_i:

$$\begin{aligned} P(\beta_i|\gamma_1) &= \int_0^\infty P(\beta_i|\tau_i) P(\tau_i|\gamma_1) d\tau_i \\ &= \frac{\sqrt{\gamma_1}}{2} \exp(-\sqrt{\gamma_1}\,|\beta_i|) \end{aligned} \quad (12)$$

which shows that a Laplacian prior is equivalent to a two-level hierarchical model characterized by zero-mean Gaussian priors with independent variances and an exponential hyperprior for those variances.

In estimating the parameter θ_i, we can adopt a prior that is similar to that for β_i but differs in one critical aspect: we must ensure that our algorithms learn $\theta_i \geqslant 0$. The reason for this requirement is somewhat subtle. Essentially, θ_i measures the scaling of the individual genes. In the forms of the kernels that we have described in equation (8), using a negative scaling θ_i effectively implies that if we compare two gene expression profiles using these kernels, similar levels of expression of that particular gene would actually reduce the value of that kernel function between the two expression profiles. Though greater similarity of a particular gene's expression levels in two different expression profiles need not necessarily imply that these two expression profiles are *more* similar (in the context of diagnostic classification, especially when the particular gene is irrelevant for the diagnosis), it can never imply that the profiles are somehow *less* similar. Thus, even though θ_i can be exactly zero, it can never be negative. As a result, we consider a two-level hierarchical model for θ_i that explicitly makes them non-negative. In particular, we adopt non-negative Gaussian priors on $\boldsymbol{\theta}$ with independent variances given by ρ_i, and exponential priors on

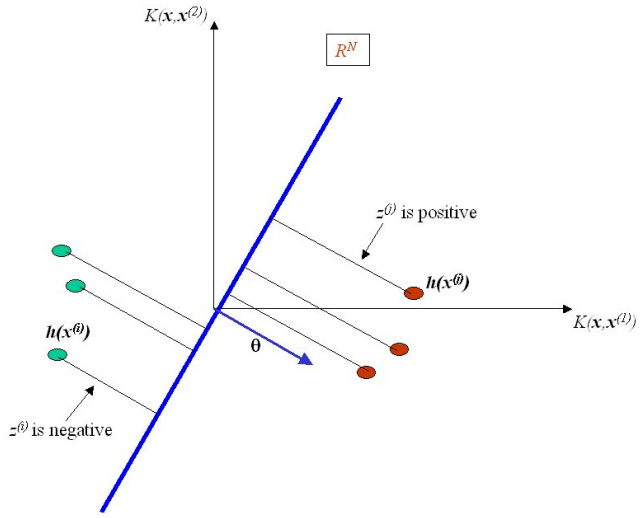

Figure 3: Latent variable interpretation of z: the variable z represents the distance in kernel space between the sample and the hyperplane describing the classifier (parameterized by $\boldsymbol{\theta}$). The probit link function provides the probability of belonging to a particular class, so class membership is determined by the sign of z.

ρ_i. This is described below:

$$P(\theta_i|\rho_i) = \begin{cases} N(\theta_i|0,\rho_i) & \text{if } \theta_i \geqslant 0 \\ 0 & \text{if } \theta_i < 0 \end{cases} \tag{13}$$

$$P(\rho_i|\gamma_2) = \frac{\gamma_2}{2}\exp\left(-\frac{\gamma_2\rho_i}{2}\right), \text{ for } \rho_i \geqslant 0 \tag{14}$$

Thus the effective prior on θ_i is:

$$P(\theta_i|\gamma_2) = \begin{cases} \sqrt{\gamma_2}\exp(-\sqrt{\gamma_2}\theta_i) & \text{if } \theta_i \geqslant 0 \\ 0 & \text{if } \theta_i < 0 \end{cases} \tag{15}$$

3.1 EM estimation of MAP parameters

Having specified the priors on the parameters that we seek to estimate, we can now proceed to the description of an EM algorithm that finds a (possibly local) maximum of the posterior distribution over $\boldsymbol{\theta}$, $\boldsymbol{\beta}$. To motivate the development of the algorithm let us consider the latent variable interpretation of the probit link function, as exploited by Albert and Chib [1], and subsequently by Figuiredo and Jain [6].

Let $z(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\beta}) = \boldsymbol{\beta}^T\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}) + w$, where w is a zero-mean unit-variance Gaussian random variable. If the classifier is defined as $y = 1$ for $z(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\beta}) \geqslant 0$ and $y = 0$ for $z(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\beta}) < 0$, then we recover the probit model, because:

$$\begin{aligned} P(y=1|\boldsymbol{x}) &= P\left(\boldsymbol{\beta}^T\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}) + w > 0\right) \\ &= \Phi\left(\boldsymbol{\beta}^T\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \end{aligned} \tag{16}$$

Figure 3 provides a graphical depiction of the intuition behind this interpretation.

Given data $D = \left\{\langle \boldsymbol{x}^{(i)}, y^i\rangle : \boldsymbol{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \{0,1\}\right\}_{i=1}^N$, consider the corresponding vector of missing variables $\boldsymbol{z} = [z^{(1)}, z^{(2)}, \ldots, z^{(N)}]^T$, as well as the vectors of missing variables $\boldsymbol{\tau} = [\tau_1, \tau_2, \ldots, \tau_M]^T$ and $\boldsymbol{\rho} = [\rho_1, \rho_2, \ldots, \rho_d]^T$. If we knew the values of $\boldsymbol{z}$, $\boldsymbol{\tau}$, and $\boldsymbol{\rho}$, we would have an easier estimation problem for $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ since we would effectively only have to find the maximum *a posteriori* solution for the following system of equations under Gaussian priors:

$$\boldsymbol{z} = \boldsymbol{H}_{\boldsymbol{\theta}}\boldsymbol{\beta} + \boldsymbol{w} \tag{17}$$

where $\boldsymbol{H}_{\boldsymbol{\theta}} = \left[\boldsymbol{h}_{\boldsymbol{\theta}}^T(\boldsymbol{x}^{(1)}), \boldsymbol{h}_{\boldsymbol{\theta}}^T(\boldsymbol{x}^{(2)}), \ldots, \boldsymbol{h}_{\boldsymbol{\theta}}^T(\boldsymbol{x}^{(N)})\right]^T$ is known as the design matrix and $\boldsymbol{w}$ is a vector of i.i.d. zero-mean unit-variance Gaussian samples. This suggests the use of an EM algorithm to find a locally maximum *a posteriori* estimate of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$. We consider $\boldsymbol{\tau}$, $\boldsymbol{\rho}$, and $\boldsymbol{z}$ as hidden variables and $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ as the parameters to be estimated. The EM algorithm will produce a sequence of estimates for $\hat{\boldsymbol{\beta}}^{(t)}$ and $\hat{\boldsymbol{\theta}}^{(t)}$ by alternating between two steps:

E-step: The log-posterior on the parameters that we seek to estimate (here $\boldsymbol{\beta}, \boldsymbol{\theta}$) given the data D and the hidden variables (here $\boldsymbol{\tau}$, $\boldsymbol{\rho}$, and $\boldsymbol{z}$), is $\log(P(\boldsymbol{\beta},\boldsymbol{\theta}|\boldsymbol{z},\boldsymbol{\tau},\boldsymbol{\rho},\boldsymbol{y}))$. In the E-step we compute the expected value of this log-posterior conditioned on the data D and the current estimate of the parameters, $\hat{\boldsymbol{\beta}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)}$. This is usually denoted as the Q function:

$$\begin{aligned} Q\left(\boldsymbol{\beta},\boldsymbol{\theta} \,\middle|\, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\beta}}^{(t)}\right) = \int P\left(\boldsymbol{z},\boldsymbol{\tau},\boldsymbol{\rho} \,\middle|\, \boldsymbol{y}, \hat{\boldsymbol{\beta}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)}\right) \times \\ \log P(\boldsymbol{\beta},\boldsymbol{\theta}|\boldsymbol{z},\boldsymbol{\tau},\boldsymbol{\rho},\boldsymbol{y}) d\boldsymbol{z} d\boldsymbol{\tau} d\boldsymbol{\rho} \end{aligned} \tag{18}$$

M-step: Update the current parameter estimate according to:

$$\hat{\boldsymbol{\theta}}^{(t+1)}, \hat{\boldsymbol{\beta}}^{(t+1)} = \arg\max_{\boldsymbol{\beta},\boldsymbol{\theta}} Q\left(\boldsymbol{\beta},\boldsymbol{\theta} \,\middle|\, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\beta}}^{(t)}\right) \tag{19}$$

In what follows, we drop the $\boldsymbol{\theta}$ subscripts on the $\boldsymbol{H}$ and $\boldsymbol{h}$ terms to simplify the notation. As a first step, we see that the complete log-posterior on the learning parameters $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$, including the hidden variables $\boldsymbol{\tau}$, $\boldsymbol{\rho}$, and $\boldsymbol{z}$, is:

$$\begin{aligned} &\log P(\boldsymbol{\beta},\boldsymbol{\theta}|\boldsymbol{y},\boldsymbol{z},\boldsymbol{\rho},\boldsymbol{\tau}) \\ &\quad\propto \log P(\boldsymbol{z}|\boldsymbol{\beta},\boldsymbol{\theta}) + \log P(\boldsymbol{\beta}|\boldsymbol{\tau}) + \log P(\boldsymbol{\theta}|\boldsymbol{\rho}) \\ &\quad\propto -\|\boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{z}\|^2 - \boldsymbol{\beta}^T\boldsymbol{T}\boldsymbol{\beta} - \boldsymbol{\theta}^T\boldsymbol{R}\boldsymbol{\theta} \\ &\quad\propto -\boldsymbol{z}^T\boldsymbol{z} - \boldsymbol{\beta}^T\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{\beta} - 2\boldsymbol{z}) - \boldsymbol{\beta}^T\boldsymbol{T}\boldsymbol{\beta} - \boldsymbol{\theta}^T\boldsymbol{R}\boldsymbol{\theta} \end{aligned} \tag{20}$$

where the matrix $\boldsymbol{T} = diag(\tau_1^{-1}, \tau_2^{-1}, \ldots, \tau_M^{-1})$ and the matrix $\boldsymbol{R} = diag(\rho_1^{-1}, \rho_2^{-1}, \ldots, \rho_d^{-1})$. Thus, the Q function is:

$$\begin{aligned} Q\left(\boldsymbol{\beta},\boldsymbol{\theta} \,\middle|\, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\beta}}^{(t)}\right) = E\Big[-\boldsymbol{z}^T\boldsymbol{z} - \boldsymbol{\beta}^T\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{\beta} - 2\boldsymbol{z}) \\ - \boldsymbol{\beta}^T\boldsymbol{T}\boldsymbol{\beta} - \boldsymbol{\theta}^T\boldsymbol{R}\boldsymbol{\theta} \,\Big|\, \boldsymbol{y}, \hat{\boldsymbol{\beta}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)}\Big] \end{aligned} \tag{21}$$

Since we seek to maximize the Q function w.r.t. $\boldsymbol{\beta}$ in the EM algorithm, terms like $E\left[-\boldsymbol{z}^T\boldsymbol{z} \,\middle|\, \boldsymbol{y}, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\beta}}^{(t)}\right]$ that do not involve $\boldsymbol{\beta}$ or $\boldsymbol{\theta}$ can be effectively ignored in the M-step, and thus are irrelevant in the E-step as well. Therefore, the Q function simplifies to:

$$\begin{aligned} Q\left(\boldsymbol{\beta},\boldsymbol{\theta} \,\middle|\, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\beta}}^{(t)}\right) = -\boldsymbol{\beta}^T\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{\beta} \\ + 2\boldsymbol{\beta}^T\boldsymbol{H}^T E\left[\boldsymbol{z} \,\middle|\, \boldsymbol{y}, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\beta}}^{(t)}\right] \\ - \boldsymbol{\beta}^T E\left[\boldsymbol{T} \,\middle|\, \boldsymbol{y}, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\beta}}^{(t)}\right]\boldsymbol{\beta} \\ - \boldsymbol{\theta}^T E\left[\boldsymbol{R} \,\middle|\, \boldsymbol{y}, \hat{\boldsymbol{\theta}}^{(t)}, \hat{\boldsymbol{\beta}}^{(t)}\right]\boldsymbol{\theta} \end{aligned} \tag{22}$$

The E-step thus simplifies to computing the expectations associated with each of these terms. Fortunately, each of these computations can be expressed in closed form. As for the term associated with the expectation of $\boldsymbol{z}$, we have:

$$v_i = E\left[z^{(i)} \;\middle|\; \boldsymbol{y}, \hat{\boldsymbol{\beta}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)}\right] = \begin{cases} \boldsymbol{h}^T(\boldsymbol{x}^{(i)})\hat{\boldsymbol{\beta}}^{(t)} + \frac{N\left(\boldsymbol{h}^T(\boldsymbol{x}^{(i)})\hat{\boldsymbol{\beta}}^{(t)}\middle|0,1\right)}{1-\Phi\left(-\boldsymbol{h}^T(\boldsymbol{x}^{(i)})\hat{\boldsymbol{\beta}}^{(t)}\right)}, & \text{if } y^{(i)}=1 \\ \boldsymbol{h}^T(\boldsymbol{x}^{(i)})\hat{\boldsymbol{\beta}}^{(t)} - \frac{N\left(\boldsymbol{h}^T(\boldsymbol{x}^{(i)})\hat{\boldsymbol{\beta}}^{(t)}\middle|0,1\right)}{\Phi\left(-\boldsymbol{h}^T(\boldsymbol{x}^{(i)})\hat{\boldsymbol{\beta}}^{(t)}\right)}, & \text{if } y^{(i)}=0 \end{cases} \quad (23)$$

which follows from the observation that $z^{(i)}$ is distributed as a Gaussian with mean $\boldsymbol{h}^T(\boldsymbol{x}^{(i)})\hat{\boldsymbol{\beta}}^{(t)}$, but left-truncated at zero if $y^{(i)}=1$, and right-truncated at zero if $y^{(i)}=0$. After some further algebraic manipulations, it can be shown that for the term associated with the expectation of $\boldsymbol{T}$, we have:

$$\omega_i = E\left[\tau_i^{-1} \;\middle|\; \boldsymbol{y}, \hat{\boldsymbol{\beta}}_i^{(t)}, \gamma_1\right] = \frac{\int_0^\infty \tau_i^{-1} P(\tau_i|\gamma_1) P\left(\hat{\boldsymbol{\beta}}_i^{(t)} \;\middle|\; \tau_i\right) d\tau_i}{\int_0^\infty P(\tau_i|\gamma_1) P\left(\hat{\boldsymbol{\beta}}_i^{(t)} \;\middle|\; \tau_i\right) d\tau_i} = \gamma_1 \left|\hat{\boldsymbol{\beta}}_i^{(t)}\right|^{-1} \quad (24)$$

The last term in the E-step computation is associated with the expectation of $\boldsymbol{R}$, and a manipulation similar to that above yields the following:

$$\delta_i = E\left[\rho_i^{-1} \;\middle|\; \boldsymbol{y}, \hat{\boldsymbol{\beta}}^{(t)}, \hat{\boldsymbol{\theta}}_i^{(t)}, \gamma_2\right] = \gamma_2 \left(\hat{\boldsymbol{\theta}}_i^{(t)}\right)^{-1} \quad (25)$$

Note that our prior on $\boldsymbol{\theta}$ requires that each θ_i be non-negative, so we employ a constrained optimization here to ensure that this remains true. If θ_i becomes exactly zero for some values of i, we can simply prune those features and continue with the remainder of the optimization.

In summary, all three integrations required for the expectation terms in the E-step can be done analytically. If we define $\boldsymbol{v} = [v_1, v_2, \ldots, v_N]^T$, $\boldsymbol{\Omega} = diag(\omega_1, \omega_2, \ldots, \omega_M)$ and $\boldsymbol{\Delta} = diag(\delta_1, \delta_2, \ldots, \delta_d)$, then in the M-step we have to maximize the following Q function with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ jointly:

$$Q\left(\boldsymbol{\beta}, \boldsymbol{\theta} \;\middle|\; \hat{\boldsymbol{\beta}}^{(t)}, \hat{\boldsymbol{\theta}}^{(t)}\right) = -\boldsymbol{\beta}^T\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{\beta} + 2\boldsymbol{\beta}^T\boldsymbol{H}^T\boldsymbol{v} - \boldsymbol{\beta}^T\boldsymbol{\Omega}\boldsymbol{\beta} - \boldsymbol{\theta}^T\boldsymbol{\Delta}\boldsymbol{\theta} \quad (26)$$

$$\frac{\partial Q}{\partial \boldsymbol{\beta}} = 2\boldsymbol{H}^T\boldsymbol{v} - 2\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{\beta} - 2\boldsymbol{\Omega}\boldsymbol{\beta} \quad (27)$$

$$\frac{\partial Q}{\partial \theta_l} = -2\delta_l\theta_l - 2\sum_{n=1}^{N}\sum_{m=1}^{M}(\boldsymbol{H}\boldsymbol{\beta} - \boldsymbol{v})\boldsymbol{\beta}^T \otimes \left(\frac{\partial \boldsymbol{H}}{\partial \theta_l}\right) \quad (28)$$

where $\otimes$ represents the element-wise matrix Hadamard product. In this paper, we have primarily used polynomial kernel classifiers of the form given in equation (8) so that $H_{i,1}=1$, and $H_{i,(j+1)} = (1+(\boldsymbol{x}^{(i)})^T\boldsymbol{\theta}\boldsymbol{x}^{(j)})$. This means that $\frac{\partial H_{i,1}}{\partial \theta_l} = 0$, and $\frac{\partial H_{i,(j+1)}}{\partial \theta_l} = n(1+(\boldsymbol{x}^{(i)})^T\boldsymbol{\theta}(\boldsymbol{x}^{(j)}))^{n-1}x_l^{(i)}x_l^{(j)}$ for $j=1,2,\ldots,N$.

Since $\boldsymbol{H}$ is in general a nonlinear function of $\boldsymbol{\theta}$, Q is also highly nonlinear and cannot be maximized analytically. Moreover, the optimization of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ cannot be pursued independently. However, we can exploit the fact that for any given $\boldsymbol{\theta}$, the optimal $\boldsymbol{\beta}_{\boldsymbol{\theta}}$ corresponding to it can be evaluated analytically by setting $\frac{\partial Q}{\partial \beta} = 0$ above. Thus, we have:

$$\hat{\boldsymbol{\beta}_{\boldsymbol{\theta}}}^{(t+1)} = (\boldsymbol{\Omega} + \boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}\boldsymbol{v} = \boldsymbol{\kappa}(\boldsymbol{I} + \boldsymbol{\kappa}\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{\kappa})^{-1}\boldsymbol{\kappa}\boldsymbol{H}\boldsymbol{v} \quad (29)$$

where $\boldsymbol{\kappa} = diag(k_1, k_2, \ldots, k_M)$ and its diagonal entries are $k_i = \omega_i^{-1/2} = \gamma_1^{-1/2}\left|\hat{\boldsymbol{\beta}}_i^{(t)}\right|^{1/2}$. The matrix $\boldsymbol{\kappa}$ has been introduced to enable a stable numerical implementation, which is necessary since the sparsity-promoting properties of the hierarchical priors will drive several of the β_i to zero, thereby causing numerical instabilities in any implementation using $\boldsymbol{\Omega}$ directly.

Although $\hat{\boldsymbol{\beta}}^{(t+1)}$ can be computed straightforwardly, we are forced to employ numerical nonlinear optimization techniques to obtain $\hat{\boldsymbol{\theta}}^{(t+1)}$ from the M-step. In our research, we have used the implementation of a subspace trust region method that is contained in the Matlab optimization toolbox and is based on the interior-reflective Newton method of Coleman and Li [4]. This is an iterative method, each iteration of which involves the approximate solution of a large linear system using the method of preconditioned conjugate gradients. This method also requires that we provide the derivatives of the Q function with respect to $\boldsymbol{\theta}$, which we have computed in equation (28).

We summarize the full algorithm in the subsection below for clarity.

3.2 Summary of the JCFO algorithm

1. Given the training set D, use an initial uninformative scaling of $\theta_i = 1$ for all the features to compute the initial design matrix $\boldsymbol{H}$.
2. Using the initial design matrix $\boldsymbol{H}$, compute an initial seed estimate for $\boldsymbol{\beta}$ using a weakly-penalized ridge regression with the labels as data. In other words, compute $\boldsymbol{\beta} = (\varepsilon\boldsymbol{I} + \boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}\boldsymbol{y}$ with a suitably small ε (we have used 10^{-6} in our experiments). Note that this corresponds to a weak zero-mean Gaussian prior with a very large variance for each of the elements of $\boldsymbol{\beta}$.
3. Using the initial seed estimate for $\boldsymbol{\beta}$ and the initial uninformative scaling $\boldsymbol{\theta}$, compute the priors $\boldsymbol{\Omega}$ and $\boldsymbol{\Delta}$ using equations (24) and (25), respectively.
4. With the above $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ as a starting point, use a nonlinear optimization technique to find the values of $\hat{\boldsymbol{\beta}}^{(0)}$ and $\hat{\boldsymbol{\theta}}^{(0)}$ that maximize (26). This provides a suitable starting point for the EM algorithm below.
5. *E-Step*: Given the current estimates for $\hat{\boldsymbol{\beta}}^{(t)}$ and $\hat{\boldsymbol{\theta}}^{(t)}$, update the values of $\boldsymbol{v}$, $\boldsymbol{\Omega}$, and $\boldsymbol{\Delta}$ using equations (23), (24), and (25), respectively.
6. *M-Step*: Obtain the new estimates $\hat{\boldsymbol{\beta}}^{(t+1)}$ and $\hat{\boldsymbol{\theta}}^{(t+1)}$ using the values of $\boldsymbol{v}$, $\boldsymbol{\Omega}$, and $\boldsymbol{\Delta}$ from the E-step by maximizing (26).
7. Repeat steps 5 and 6 until convergence of the log-likelihood.

Note that we must employ a constrained optimization in steps 4 and 6 to ensure satisfaction of the linear inequality

constraints $\theta_i \geqslant 0$. Also note that if we use a linear classifier with the formulation for $\boldsymbol{h}$ given in equation (5), then we have a fixed $\boldsymbol{\theta}$ and consequently, our M-step simply reduces to equation (29). In this case, no nonlinear optimization would be required, but the sparsity on $\boldsymbol{\beta}$ would still ensure that we use the expression levels of very few genes in designing a hyperplane classifier. We have used this version as a fast but powerful feature selection algorithm to reduce the dimensionality of the data that has to be handled by the JCFO with a linear (i.e., polynomial order 1) kernel. Thus, we can first perform the feature selection using exactly the same algorithm by simply changing our design matrix and keeping $\boldsymbol{\theta}$ constant; this procedure typically identifies about 50 genes to be of relevance (non-zero scaling). Then, we can quickly run the JCFO since the dimensionality of the search space for the nonlinear optimization in the M-step is much smaller (and thus the search is much faster).

It should be observed that although both the JCFO with a linear kernel and the simpler version above using (5) construct sparse hyperplane classifiers, they will be expected to perform differently due to the different priors on each model. Using (5), we have sparsity on the features but not on the kernel coefficients, while the JCFO still has both kinds of sparsity. Since this affords an extra amount of regularization, we have observed greater classification accuracy for the JCFO, even though both methods construct hyperplane classifiers.

4. EXPERIMENTAL RESULTS

In this paper, we are the first to apply the sparse probit regression algorithm of Figuiredo and Jain to the problem of classifying gene expression data, but our primary contribution is the development of the JCFO algorithm. To gauge the efficacy of both of these algorithms, we tested them on two benchmark datasets; both the datasets provide expression levels for human genes produced by Affymetrix high-density oligonucleotide microarrays. To measure diagnostic accuracy, we use a full leave-one-out cross validation procedure (LOOCV) where we train on $N-1$ samples and test on the remaining sample which has not been used during training. By cycling through all the samples, we can get realistic estimates of the efficacy of these methods. In all our experiments, we normalize the expression levels for each gene by subtracting the mean and dividing by the standard deviation of that gene. In addition, in order to reduce the computational cost of the full LOOCV procedure, we accelerated our feature selection by preprocessing using the JCFO algorithm with a linear $\boldsymbol{h}$ function of the form given in equation (5) as explained above. Thus, we were able to perform the full LOOCV error rate estimation on an 800MHz Pentium III Windows machine within a couple of hours, by taking advantage of the reduction in dimensionality of the search space for the general nonlinear optimization. It is worth pointing out that we did *not* preselect features first by using a LOOCV, and then perform the classification. Instead the above two-step process was simply a proxy for a single, larger optimization problem.

We used two benchmark datasets reported in the literature for this problem. The first dataset contains examples of human acute leukemia, originally analyzed by Golub, et al. [9]. The dataset containing expression levels of 7129 genes can be obtained at http://www-genome.wi.mit.edu/mpr/table_AML_ALL_samples.rtf. To collect this data, bone marrow or blood samples were taken from 72 patients, 47 with acute myeloid leukemia (AML) and 25 with acute lymphoblastic leukemia (ALL). The second dataset contains expression levels of 2000 genes from 40 tumor and 22 normal colon tissues. The dataset was originally analyzed by Alon, et al. [2] and is available at http://www.molbio.princeton.edu/colondata.

Table 1: Accuracy of diagnostic classification: multiple classifiers applied to two benchmark datasets (% correct in LOOCV study)

Classifier	AML/ ALL	Colon tumor
Adaboosting (Decision stumps) [3]	95.8	72.6
SVM (Linear kernel) [3]	94.4	77.4
SVM (Quadratic kernel) [3]	95.8	74.2
RVM (No kernel: on feature space) [10]	97.2	88.7
Logistic regression (No kernel: on feature space) [10]	97.2	71.0
Sparse probit regression (Quadratic kernel)	95.8	84.6
Sparse probit regression (Linear kernel)	97.2	91.9
JCFO (Quadratic kernel)	98.6[1]	88.7
JCFO (Linear kernel)	100.0	96.8

[1] For this one classification task, we preselected 100 genes using FDR (the Fisher discriminant ratio) instead of preprocessing the genes using the linear $\boldsymbol{h}$ function as described in the text.

For all our experiments, we used polynomial kernels with the JCFO. In Table 1, we use a full leave-one-out cross-validation study for each of the two datasets to compare the accuracy of the diagnostic classification reported by the JCFO against that reported by Adaboosting, the SVM, the RVM, logistic regression, and sparse probit regression. Since our classifier has been designed to identify the optimal genes as well as the optimal classifier (i.e., $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$), we further analyzed the genes that are identified as most important by the classifier. Tables 2 and 3 show the genes identified by the JCFO as being most important for making a diagnostic decision. The reported values of $\boldsymbol{\theta}$ in these tables were obtained by taking the mean of the $\boldsymbol{\theta}$ obtained for each of the classifiers designed in the LOOCV (so $N = 72$ in Table 2 and $N = 62$ in Table 3).

5. DISCUSSION

First, as indicated in Table 1, the classification accuracy of the JCFO is superior to all of the other classification methods we tested or found in the literature, including the sparse probit regression EM algorithm with a Jeffereys prior as proposed by Figuiredo and Jain. Specifically, the Jeffer-

eys EM algorithm failed to consistently learn effective classifiers when using a second-order polynomial kernel. Though the use of the Laplacian prior did permit the design of a reasonable classifier, the real benefits of nonlinear classifier design are impossible to achieve with a poorly designed polynomial kernel that weights all the features as equally relevant in measuring similarity between two sets of expression profiles of genes. The JCFO improves classification performance by finding the appropriate scaling and selection of genes to use, all as part of the algorithm. Of course, this joint optimization comes at a price: the algorithm is significantly slower than the sparse probit regression algorithm of Figuiredo and Jain, though it was still acceptable for our purposes. However, the computational complexity of our algorithm might render it impractical with current technology if it were applied to datasets with several hundreds or thousands of samples. Fortunately, the sample sizes available in current datasets are typically much smaller. The large number of genes does not pose a problem since we preselect all the genes found useful by the sparse probit regression algorithm, and use our algorithm only on this reduced set of around fifty genes. In this case, our algorithm converges in less than half an hour.

Second, the genes that the JCFO algorithm associates with high $\boldsymbol{\theta}$ values for each of these cancer types are shown in Tables 2 and 3. We note that almost all genes with high values of $\boldsymbol{\theta}$ in Table 2 are of known relevance to the AML/ALL distinction. In particular, CST3, CD33, DF, HOXA9, LTC4S, PRG1, CTSD, and EPB72 were all determined by Golub, et al., to be predictive of AML. In addition, MPO (not identified by Golub, et al.) is known to occur in virtually all cells of the myeloid lineage and none of the lymphoid lineage, and antibodies to MPO are used as clinical determinants of AML. CD33 is similarly a marker for AML, expressed in nearly all malignant myeloblasts. HOXA9 is transformed in myeloid cells and can lead to leukemia in animal models. DF (adipsin) is expressed during myeloid cell differentiation. Many others of the genes with high $\boldsymbol{\theta}$ are known to play a role in myeloid/lymphoid differentiation, and a few novel genes have been identified as well. Similar results hold in the case of the colon data. The genes with high values of $\boldsymbol{\theta}$ in Table 3 according to the JCFO are in many cases also known to be implicated in colon cancer or other cancers, including CDC42, MXI1, RNASE1, GUCA2B, REL, FGFR2, and PTPRJ. A few anomalous genes like AMH and SEMG2 are also given non-zero values of θ, and we are currently investigating these genes for novel properties.

In conclusion, the JCFO has successfully achieved both of its objectives: LOOCV classification accuracy above the state-of-the-art and automatic gene selection. Nevertheless, considerable scope for improvement remains and we must still address several questions. Is our prior optimal? Does our EM formulation converge to a global maximum or do we face multiple local maxima? We are investigating different ways to address these issues in our current work. We have recently compared the JCFO to a number of other methods on a much larger collection of datasets; results from this comparison are forthcoming [11]. All code developed as part of this research is available from the first author.

6. REFERENCES

[1] J. Albert and S. Chib. Bayesian analysis of binary and polychotomous response data. *J. of the Amer. Stat. Assoc*, 88:669–679, 1993.

[2] U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, and A.J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA*, 96:6745–6750, 1999.

[3] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini. Tissue classification with gene expression profiles. In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology (RECOMB)*. Universal Academy Press, Tokyo, Apr. 2000.

[4] T.F. Coleman and Y. Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6:418–445, 1996.

[5] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and other kernel based learning methods)*. Cambridge University Press, 2000.

[6] M.A.T. Figuiredo and A.K. Jain. Bayesian learning of sparse classifiers. In *2001 Conference on Computer Vision and Pattern Recognition (CVPR 2001)*. IEEE Press, Dec. 2001.

[7] A. Frank. A new branch and bound feature selection algorithm. *M.Sc. Thesis*, submitted to Technion, Israel Institute of Technology, 2002.

[8] T.S. Furey, N. Cristianini, N. Duffy, D.W. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[9] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caliguiri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[10] B. Krishnapuram, A.J. Hartemink, and L. Carin. Logistic regression and RVM for cancer diagnosis from gene expression signatures. In *Proceedings of the 2002 Workshop on Genomic Signal Processing and Statistics (GENSIPS)*. IEEE Signal Processing Society, Oct. 2002.

[11] B. Krishnapuram, P. Pratapa, L. Carin, and A.J. Hartemink. A comprehensive comparison of sparse Bayesian methods for disease diagnosis based on gene expression. (in preparation, 2003).

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

[13] M.E. Tipping. Sparse Bayesian learning and the relevance vector machine. In *Journal of Machine Learning Research*, 1:211–244, 2001.

[14] M. Xiong, W. Li, J. Zhao, L. Jin, and E. Boerwinkle. Feature (gene) selection in gene expression-based tumor classification. *Molecular Genetics and Metabolism*, 73:239–247, 2001.

Table 2: Most important genes for distinguishing between AML and ALL, as selected by the JCFO

θ_i	Index	Accession	Gene Name	Gene Description
1.14	1780	M19507	MPO	myeloperoxidase
0.83	3848	U82759	HOXA9	homeo box A9
0.81	1797	M20902	APOC1	apolipoprotein C-I
0.77	1830	M22960	PPGB	protective protein for beta-galactosidase (galactosialidosis)
0.70	4952	Y07604	NME4	non-metastatic cells 4, protein expressed in
0.67	5599	L15326	PTGS2	prostaglandin-endoperoxide synthase 2
0.56	5003	Y10207	CD171	Human CD171 protein
0.51	5108	Z29067	NEK3	NIMA (never in mitosis gene a)-related kinase 3
0.46	1883	M27891	CST3	cystatin C (amyloid angiopathy and cerebral hemorrhage)
0.42	6540	X85116	EPB72	erythrocyte membrane protein band 7.2 (stomatin)
0.42	2289	M84526	DF	D component of complement (adipsin)
0.41	6185	M14483	PTMA	prothymosin, alpha (gene sequence 28)
0.41	879	Y00371	HSPA8	heat shock 70kDa protein 8
0.40	5349	M61853	CYP2C18	cytochrome P450, subfamily IIC, polypeptide 18
0.35	1835	M23197	CD33	CD33 antigen (gp67)
0.34	4197	X17042	PRG1	proteoglycan 1, secretory granule
0.33	6170	M13690	SERPING1	serine (or cysteine) proteinase inhibitor, clade G, member 1
0.32	1395	L20941	FTH1	ferritin, heavy polypeptide 1
0.30	1942	M31994	ALDH1A1	aldehyde dehydrogenase 1 family, member A1
0.29	3321	U50136	LTC4S	leukotriene C4 synthase
0.27	5767	X13294	MYBL1	v-myb myeloblastosis viral oncogene homolog (avian)-like 1
0.26	1976	J02963	ITGA2B	integrin, alpha 2b (platelet glycoprotein IIb, antigen CD41B)
0.23	805	HG1612	MACMARCKS	macrophage myristoylated alanine-rich C kinase substrate
0.18	6056	U28055	MST1	macrophage stimulating 1 (hepatocyte growth factor-like)
0.16	2122	M63138	CTSD	cathepsin D (lysosomal aspartyl protease)
0.16	1934	M31627	XBP1	X-box binding protein 1
0.12	3392	U53468	NDUFA5	NADH dehydrogenase (ubiquinone) 1 alpha subcomplex, 5, 13kDa
0.08	3715	U77604	MGST2	microsomal glutathione S-transferase 2
0.07	6226	M28170	CD19	CD19 antigen
0.03	1686	M11722	DNTT	deoxynucleotidyltransferase, terminal

Table 3: Most important genes for distinguishing colon cancer, as selected by the JCFO

θ_i	Index	Accession	Gene Name	Gene Description
2.10	1357	T84051	CDC42	cell division cycle 42 (GTP binding protein, 25kDa)
1.76	974	U00968	SREBF1	sterol regulatory element binding transcription factor 1
1.47	1924	H64807		placental folate transporter (H. sapiens)
1.44	1873	L07648	MXI1	MAX interacting protein 1
1.41	350	D26129	RNASE1	ribonuclease, RNase A family, 1 (pancreatic)
1.38	377	Z50753	GUCA2B	guanylate cyclase activator 2B (uroguanylin)
1.21	1757	H16096	PMPCB	peptidase (mitochondrial processing) beta
1.01	765	M76378	CSRP1	cysteine and glycine-rich protein 1
0.86	1346	T62947	RPL24	ribosomal protein L24
0.84	1976	K03474	AMH	anti-Mullerian hormone
0.75	792	R88740	ATP5J	ATP synthase, H+ transporting, mitochondrial F0 complex, subunit F6
0.74	70	T61661	PFN1	profilin 1
0.74	554	H24401		MAP kinase phosphatase-1 (H. sapiens)
0.74	698	T51261	SERPINE2	serine (or cysteine) proteinase inhibitor, clade E (nexin), member 2
0.72	1546	T51493	PPP2R5C	protein phosphatase 2, regulatory subunit B (B56), gamma isoform
0.64	1740	M81651	SEMG2	semenogelin II
0.50	1641	K02268	PDYN	prodynorphin
0.42	1024	R65697	REL	v-rel reticuloendotheliosis viral oncogene homolog (avian)
0.37	1644	R80427		C4-dicarboxylate transport sensor protein DCTB (R. leguminosarum)
0.32	1623	T94993	FGFR2	fibroblast growth factor receptor 2 (keratinocyte growth factor receptor)
0.14	1909	U10886	PTPRJ	protein tyrosine phosphatase, receptor type, J
0.12	1482	T64012	CHRND	cholinergic receptor, nicotinic, delta polypeptide
0.10	1094	R33481	ATF7	activating transcription factor 7
0.06	187	T51023	HSPCB	heat shock 90kDa protein 1, beta
0.06	1504	H78386	IL1R2	interleukin 1 receptor, type II
0.03	1241	T64885		general negative regulator of transcription subunit 1 (S. cerevisiae)