Probabilistic Feature Selection and Classification Vector Machine

BINGBING JIANG, University of Science and Technology of China
CHANG LI and MAARTEN DE RIJKE, University of Amsterdam
XIN YAO, Southern University of Science and Technology
HUANHUAN CHEN, University of Science and Technology of China

Sparse Bayesian learning is a state-of-the-art supervised learning algorithm that can choose a subset of relevant samples from the input data and make reliable probabilistic predictions. However, in the presence of high-dimensional data with irrelevant features, traditional sparse Bayesian classifiers suffer from performance degradation and low efficiency due to the incapability of eliminating irrelevant features. To tackle this problem, we propose a novel sparse Bayesian embedded feature selection algorithm that adopts truncated Gaussian distributions as both sample and feature priors. The proposed algorithm, called probabilistic feature selection and classification vector machine ($PFCVM_{LP}$) is able to simultaneously select relevant features and samples for classification tasks. In order to derive the analytical solutions, Laplace approximation is applied to compute approximate posteriors and marginal likelihoods. Finally, parameters and hyperparameters are optimized by the type-II maximum likelihood method. Experiments on three datasets validate the performance of $PFCVM_{LP}$ along two dimensions: classification performance and effectiveness for feature selection. Finally, we analyze the generalization performance and derive a generalization error bound for $PFCVM_{LP}$. By tightening the bound, the importance of feature selection is demonstrated.

CCS Concepts: • **Computing methodologies** → **Feature selection**; *Supervised learning by classification*;

Additional Key Words and Phrases: Feature selection, probabilistic classification model, sparse Bayesian learning, supervised learning, EEG emotion recognition

This work was supported in part by the National Natural Science Foundation of China (Grant nos. 91846111 and 91746209), the Science and Technology Innovation Committee Foundation of Shenzhen (Grant no. ZDSYS201703031748284), Ahold Delhaize, Amsterdam Data Science, the Bloomberg Research Grant program, the China Scholarship Council, the Criteo Faculty Research Award program, Elsevier, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the Google Faculty Research Awards program, the Microsoft Research Ph.D. program, the Netherlands Institute for Sound and Vision, the Netherlands Organisation for Scientific Research (NWO) under project nrs CI-14-25, 652.002.001, 612.001.551, 652.001.003, and Yandex. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.
Authors' addresses: B. Jiang and H. Chen (corresponding author), School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China; emails: jiangbb@mail.ustc.edu.cn, hchen@ustc.edu.cn; C. Li and M. de Rijke, Informatics Institute, University of Amsterdam, Amsterdam, the Netherlands; email: {c.li, derijke}@uva.nl; X. Yao, Department of Computer Science and Engineering, Shenzhen Key Laboratory of Computational Intelligence, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China; email: xiny@sustc.edu.cn.

1556-4681/2019/04-ART21 $15.00
https://doi.org/10.1145/3309541

ACM Reference format:
Bingbing Jiang, Chang Li, Maarten de Rijke, Xin Yao, and Huanhuan Chen. 2019. Probabilistic Feature Selection and Classification Vector Machine. *ACM Trans. Knowl. Discov. Data* 13, 2, Article 21 (April 2019), 27 pages.
https://doi.org/10.1145/3309541

1 INTRODUCTION

In supervised learning, we are given input feature vectors $\mathbf{x} = \{x_i \in \mathbb{R}^M\}_{i=1}^N$ and corresponding labels $\mathbf{y} = \{y_i\}_{i=1}^N$.[1] The goal is to predict the label of a new datum $\hat{x}$ based on the training dataset $\mathbf{S} = \{\mathbf{x}, \mathbf{y}\}$ together with other prior knowledge. For regression, we are given continuous labels $y \in \mathbb{R}$, while for classification we are given discrete labels. In this article, we focus on the binary classification case, in which $y \in \{-1, +1\}$.

Recently, learning sparseness from large-scale datasets has generated significant research interest [9, 11, 19, 29, 34]. Among the methods proposed, the support vector machine (SVM) [11], which is based on the kernel trick [46] to create a non-linear decision boundary with a small number of support vectors, is the state-of-the-art algorithm. The prediction function of SVM is a combination of basis functions:[2]

$$f(\hat{x}; \mathbf{w}) = \sum_{i=1}^{N} \phi(\hat{x}, x_i) w_i + b, \tag{1}$$

where $\phi(\cdot, \cdot)$ is the basis function, $\mathbf{w} = \{w_i\}_{i=1}^N$ are sample weights, and b is the bias.

Similar to SVM, many sparse Bayesian classifiers also use Equation (1) as their decision function; examples include the relevance vector machine (RVM) [45] and the probabilistic classification vector machine (PCVM) [8]. Unlike SVM, whose weights are determined by maximizing the decision margin and limited to hard binary classification, sparse Bayesian algorithms optimize the parameters within a maximum likelihood framework and make predictions based on the average of the prediction function over the posterior of parameters. For example, PCVM computes the maximum *a posteriori* (MAP) estimation using the expectation–maximization (EM) algorithm; RVM and efficient probabilistic classification vector machine (EPCVM) [9] compute the type-II maximum likelihood [3] to estimate the distribution of the parameters. However, these algorithms have to deal with different scales of features due to the failure to eliminate irrelevant features.

In addition to sparse Bayesian learning, parameter-free Bayesian methods that are based on the class-conditional distributions have been proposed to solve the classification task [18, 21, 27, 28]. Lanckriet et al. [28] proposed the minimax probability machine (MPM) to estimate the bound of classification accuracy by minimizing the worst error rate. To efficiently exploit structural information of data, Gu et al. [18] proposed a structural MPM (SMPM) that can produce the non-linear decision hyperplane by using the kernel trick. To exploit structural information, SMPM adopts a clustering algorithm to detect the clusters of each class and then calculates the mean and covariance matrix for each cluster. However, selecting a proper number of clusters per class is difficult for the clustering algorithm, and calculating the mean and covariance matrix for each cluster has a high computational complexity for high-dimensional data. Therefore, SMPM cannot fit different scales of features and might suffer from the instability and low efficiency especially for high-dimensional data.

[1] In this article, the subscript of a sample x, i.e., x_i, denotes the ith sample and the superscript of a sample x, i.e., x^k, denotes the kth dimension.

[2] In the rest of this article, we prefer to use the term *basis function* instead of *kernel function* because, except for SVM, the basis functions used in this article are free of Mercer's condition.

In order to fit different scales of features, basis functions are always controlled by basis parameters (or kernel parameters). For example, in LIBSVM [7] with Gaussian radial basis functions (RBF) $\phi(x, z) = \exp(-\vartheta\|x - z\|^2)$, the default ϑ is set relatively small for high-dimensional datasets and large for low-dimensional datasets. Although the use of basis parameters may help to address the curse of dimensionality [2], the performance might be degraded when there are lots of irrelevant and/or redundant features [26, 29, 37]. Parameterized basis functions are designed to deal with this problem. There are two popular basis functions that can incorporate feature parameters easily:

Gaussian RBF

$$\phi_{\boldsymbol{\theta}}(x, z) = \exp\left(-\sum_{k=1}^{M} \theta_k (x^k - z^k)^2\right), \tag{2}$$

Pth order polynomial:

$$\phi_{\boldsymbol{\theta}}(x, z) = \left(1 + \sum_{k=1}^{M} \theta_k x^k z^k\right)^P, \tag{3}$$

where the subscript denotes the corresponding index of features, and $\boldsymbol{\theta} \in \mathbb{R}^M$ are feature parameters (also called feature weights). Once a feature weight $\theta_k \to 0$,[3] the corresponding feature will not contribute to the classification.

Feature selection, as a dimensionality reduction technique, has been extensively studied in machine learning and data mining, and various feature selection algorithms have been proposed [5, 26, 29, 35, 37–39, 41, 48, 49, 51, 53–58]. Feature selection methods can be divided into three groups: *filter methods* [20, 38, 39, 41], *wrapper methods* [51], and *embedded methods* [5, 26, 29, 34, 35, 37]. Filter methods independently select the subset of features from the classifier learning. Wrapper methods consider all possible feature subsets and then select a specific subset based on its predictive power. Therefore, the feature selection stage and classification model are separated and independent in the filter and wrapper methods, and the wrapper methods might suffer from high computational complexity especially for high-dimensional data [34]. Embedded methods embed feature selection in the training process, which aims to combine the advantages of the filter and wrapper methods. As to filter methods, Peng et al. [41] proposed a minimum redundancy and maximum relevance (mRMR) method, which selects relevant features and simultaneously removes redundant features according to the mutual information. To avoid evaluating the score for each feature individually like Fisher score [14], a filter method, trace ratio criterion (TRC) [39] was designed to find the globally optimal feature subset by maximizing the subset level score. Recently, sparsity regularization in feature space has been widely applied to feature selection tasks. In [5], Bradley and Mangasarian proposed an embedded method, L_1SVM, that uses the L_1 norm to yield a sparse solution. However, the number of features selected by L_1SVM is upper bounded by the number of training samples, which limits its application on high-dimensional data. Nie et al. [38] employed joint L_{21} norm minimization on both loss function and regularization to propose a filter method, FSNM. Based on the basis functions mentioned above, Nguyen and De la Torre [37] designed an embedded feature selection model, weight SVM (WSVM), that can jointly perform feature selection and classifier construction for non-linear SVMs. However, filter methods are not able to adaptively select relevant features, i.e., they require a predefined number of selected features.

For Bayesian feature selection approaches, a joint classifier and feature optimization algorithm (JCFO) is proposed in [26]; the authors adopt a sparse Bayesian model to simultaneously perform classifier learning and feature selection. To select relevant features, JCFO introduces hierarchical sparseness promoting priors on feature weights and then employs EM and gradient-based methods

[3]Practically, lots of feature weights $\theta_k \to 0$. When a certain θ_k is smaller than a threshold, we will set it 0.

to optimize the feature weights. In order to simultaneously select relevance samples and features, Mohsenzadeh et al. [35] extend the standard RVM and then design the relevance sample feature machine (RSFM) and an incrementally learning version (IRSFM) [34] that scales the basis parameters in RVM to a vector and applies zero-mean Gaussian priors on feature weights to generate sparsity in the feature space. Li and Chen [29] propose an EM algorithm based joint feature selection strategy for probabilistic feature selection and classification vector machine (PCVM; denoted as PFCVM_{EM}), in which they add truncated Gaussian priors to features to enable PCVM to jointly select relevant samples and features. However, JCFO, PFCVM_{EM}, and RSFM use an EM algorithm to calculate a MAP point estimate of the sample and feature parameters. As pointed out by Chen et al. [9], the EM algorithm has the following limitations: first, it is sensitive to the starting points and cannot guarantee convergence to global maxima or minima; second, the EM algorithm results in a MAP point estimate, which limits to the Bayes estimator with the 0–1 loss function and cannot represent all advantages of the Bayesian framework.

JCFO, RSFM, and IRSFM adopt a zero-mean Gaussian prior distribution over sample weights, and RSFM and IRSFM also use this prior distribution over feature weights. As a result of adopting a zero-mean Gaussian prior over samples, some training samples that belong to the positive class ($y_i = +1$) will receive negative weights and vice versa; this may result in instability and degeneration in solutions [8]. Also, for RSFM and IRSFM, zero-mean Gaussian feature priors will lead to negative feature weights, which reduces the value of kernel functions for two samples when the similarity in the corresponding features is increased [26]. Finally, RSFM and IRSFM have to construct an $N \times M$ kernel matrix for each sample, which yields a space complexity of at least $O(N^2M)$ to store the designed kernel matrices.

We propose a sparse Bayesian embedded feature selection method, i.e., a Laplace approximation based feature selection PCVM method (PFCVM_{LP}) that uses the type-II maximum likelihood method to approximate a fully Bayesian estimation. In contrast to the filter methods such as mRMR [41], FSNM [38], and TRC [39], and the embedded methods such as JCFO [26], L_1SVM [5], and WSVM [37], the proposed PFCVM_{LP} method can adaptively select informative and relevant samples and features with probabilistic predictions. Moreover, PFCVM_{LP} adopts truncated Gaussian priors as both sample and feature priors, which obtains a more stable solution and avoids the negative values for sample and feature weights. We summarize the main contributions as follows:

- Unlike traditional sparse Bayesian classifiers, like PCVM and RVM, the proposed algorithm simultaneously selects the relevant features and samples, which leads to a robust classifier for high-dimensional datasets.
- Compared with PFCVM_{EM} [29], JCFO [26], and RSFM [35], PFCVM_{LP} adopts the type-II maximum likelihood [45] approach to optimize the parameters and hyperparameters, which achieves a more stable solution and might avoid the limitations caused by the EM algorithm.
- PFCVM_{LP} is extensively evaluated and compared with the state-of-the-art feature selection methods on different real-world datasets. The results validate the performances of PFCVM_{LP}.
- We derive a generalization bound for PFCVM_{LP}. By analyzing the bound, we demonstrate the significance of feature selection and introduce a way of choosing the initial values.

The rest of the article is structured as follows. Background knowledge of sparse Bayesian learning is introduced in Section 2. Section 3 details the implementation of simultaneously optimizing sample and feature weights of PFCVM_{LP}. In Section 4, experiments are designed to evaluate both the accuracy of classification and the effectiveness of feature selection. Analyses of sparsity and generalization for PFCVM_{LP} are presented in Section 5. We conclude in Section 6.

2 SPARSE BAYESIAN LEARNING FRAMEWORK

In the sparse Bayesian learning framework, we usually use the Laplace distribution and/or the student's-t distribution as the sparseness-promoting prior. In binary classification problems, we choose a Bernoulli distribution as the likelihood function. Together with the proper marginal likelihood, we can compute the parameters' distribution (posterior distribution) either by MAP point estimation or by a complete Bayesian estimation approximated by type-II maximum likelihood. Below, we detail the implementation of this framework.

2.1 Model Specification

We concentrate on a linear combination of basis functions. To simplify our notation, the decision function is defined as

$$f(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}) = \boldsymbol{\Phi}_{\boldsymbol{\theta}}(\mathbf{x})\mathbf{w}, \tag{4}$$

where $\mathbf{w}$ denotes the $N+1$-dimensional sample weights; w_0 denotes the bias; $\boldsymbol{\Phi}_{\boldsymbol{\theta}}(\mathbf{x})$ is an $N \times (N+1)$ basis function matrix, except for the first column $\boldsymbol{\phi}_{\boldsymbol{\theta},0}(\mathbf{x}) = [1, \ldots, 1]^T$, other component $\phi_{\boldsymbol{\theta},ij} = \phi_{\boldsymbol{\theta}}(x_i, x_j) \times y_j$,[4] and $\boldsymbol{\theta} \in \mathbb{R}^M$ is the feature weights.

As probabilistic outputs are continuous values in $[0, 1]$, we need a link function to obtain a smooth transformation from $[-\infty, +\infty]$ to $[0, 1]$. Here, we use a sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ to map Equation (4) to $[0, 1]$. Then, we combine this mapping with a Bernoulli distribution to compute the following likelihood function:

$$p(\mathbf{t} \mid \mathbf{w}, \boldsymbol{\theta}, \mathrm{S}) = \prod_{i=1}^{N} \sigma_i^{t_i} (1 - \sigma_i)^{(1-t_i)},$$

where $t_i = (y_i + 1)/2$ denotes the probabilistic target of the ith sample and σ_i denotes the sigmoid mapping for the ith sample: $\sigma_i = \sigma(f(x_i; \mathbf{w}, \boldsymbol{\theta}))$. The vector $\mathbf{t} = (t_1, \ldots, t_N)^T$ consists of the probabilistic targets of all training samples and $\mathrm{S} = \{\mathbf{x}, \mathbf{y}\}$ is the training set.

2.2 Priors Over Samples and Features

According to Chen et al. [8], a truncated Gaussian prior may result in the proper sparseness to sample weights. Following this idea, we introduce a non-negative left-truncated Gaussian prior $\mathcal{N}_t(w_i \mid 0, \alpha_i^{-1})$ to each sample weight w_i:

$$\begin{aligned} p(w_i \mid \alpha_i) &= \begin{cases} 2\mathcal{N}\left(w_i \mid 0, \alpha_i^{-1}\right) & \text{if } w_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \\ &= 2\mathcal{N}\left(w_i \mid 0, \alpha_i^{-1}\right) \cdot 1_{w_i \geq 0}(w_i), \end{aligned} \tag{5}$$

where α_i (precision) is a hyperparameter, which is equal to the inverse of variance, and $1_{x \geq 0}(x)$ is an indicator function that returns 1 for each $x \geq 0$ and 0 otherwise. For the bias w_0, we introduce a zero-mean Gaussian prior $\mathcal{N}(w_0 \mid 0, \alpha_0^{-1})$:

$$p(w_0 \mid \alpha_0) = \mathcal{N}\left(w_0 \mid 0, \alpha_0^{-1}\right). \tag{6}$$

Assuming that the sample weights are independent and identically distributed (i.i.d.), we can compute the priors over sample weights as follows:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=0}^{N} p(w_i|\boldsymbol{\alpha}_i) = \mathcal{N}\left(w_0|0, \alpha_0^{-1}\right) \prod_{i=1}^{N} \mathcal{N}_t\left(w_i|0, \alpha_i^{-1}\right), \tag{7}$$

[4]We assume that each sample weight has the same sign as the corresponding label. So by multiplying the basis vector with the corresponding label, we can assume that all sample weights are non-negative.

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_N)^T$ and $\mathcal{N}_t(w_i \mid 0, \alpha_i^{-1})$ denote the left truncated Gaussian distribution.

Feature weights indicate the importance of features. For important features, the corresponding weights are set to relatively large values and vice versa. For irrelevant and/or redundant features, the weights are set to 0. Following [26], we should not allow negative values for feature weights. Based on these discussions, we introduce left truncated Gaussian priors for feature weights. Under the i.i.d. assumption, the prior over features is computed as follows:

$$p(\boldsymbol{\theta} \mid \boldsymbol{\beta}) = \prod_{k=1}^{M} p(\theta_k \mid \beta_k) = \prod_{k=1}^{M} \mathcal{N}_t\left(\theta_k \mid 0, \beta_k^{-1}\right),$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_M)^T$ are hyperparameters of feature weights. Each prior is formalized as follows:

$$\begin{aligned} p(\theta_k \mid \beta_k) &= \begin{cases} 2\mathcal{N}\left(\theta_k \mid 0, \beta_k^{-1}\right) & \text{if } \theta_k \geq 0, \\ 0 & \text{otherwise,} \end{cases} \\ &= 2\mathcal{N}\left(\theta_k \mid 0, \beta_k^{-1}\right) \cdot 1_{\theta_k > 0}(\theta_k). \end{aligned} \tag{8}$$

For both kinds of priors, we introduce Gamma distributions for α_i and β_k as hyperpriors. The truncated Gaussian priors will work together with the flat Gamma hyperpriors and result in truncated hierarchical Student's-t priors over weights. These hierarchical priors, which are similar to Laplace priors, work as L1 regularization and lead to sparse solutions [8, 26].

2.3 Computing Posteriors

The posterior in a Bayesian framework contains the distribution of all parameters. Computing parameters boils down to updating posteriors. Having priors and likelihood, posteriors can be computed with the following formula:

$$p(\mathbf{w}, \boldsymbol{\theta} \mid \mathbf{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{p(\mathbf{t} \mid \mathbf{w}, \boldsymbol{\theta}, \mathrm{S}) p(\mathbf{w} \mid \boldsymbol{\alpha}) p(\boldsymbol{\theta} \mid \boldsymbol{\beta})}{p(\mathbf{t} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathrm{S})}. \tag{9}$$

Some methods, such as PCVM, $\mathrm{PFCVM}_{\mathrm{EM}}$, and JCFO, overlook information in the marginal likelihood and use the EM algorithm to obtain a MAP point estimation of parameters. Although an efficient estimation might be obtained by the EM algorithm, it overlooks the information in the marginal likelihood and is not regarded as a complete Bayesian estimation. Other methods, such as RVM and EPCVM, retain the marginal likelihood. They compute the type-II maximum likelihood and obtain a complete Bayesian solution.

The predicted distribution for the new datum $\hat{x}$ is computed as follows:

$$p(\hat{y} \mid \hat{x}, \mathbf{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \int p(\hat{y} \mid \hat{x}, \mathbf{w}, \boldsymbol{\theta}) p(\mathbf{w}, \boldsymbol{\theta} \mid \mathbf{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}) d\mathbf{w} d\boldsymbol{\theta}.$$

If both terms in the integral are Gaussian distributions, it is easy to compute this integral analytically. We will detail the implementation of $\mathrm{PFCVM}_{\mathrm{LP}}$ in the next section.

3 PROBABILISTIC FEATURE SELECTION CLASSIFICATION VECTOR MACHINE

Details of computing sample weights and sample hyperparameters were reported by Chen et al. [9]. In this section, we mainly focus on computing parameters and hyperparameters for features.

3.1 Approximations for Posterior Distributions

Since the indicator function in Equation (8) is not differentiable, an approximate function is required to smoothly approximate the indicator function. Here, we use a parameterized sigmoid

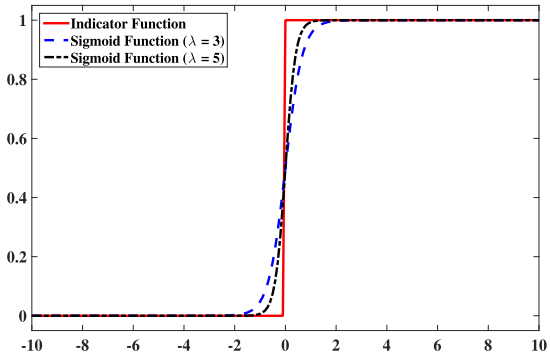


Fig. 1. Illustration of the indicator function and the sigmoid function.

assumption. Figure 1 shows the approximation of an indicator function made by a sigmoid function $\sigma(\lambda x)$. As depicted in Figure 1, the larger λ is, the more accurate approximation a sigmoid function will make. In PFCVM$_{\text{LP}}$, we choose $\sigma(5x)$ as the approximation function.

We calculate Equation (9) by the Laplace approximation, in which the Gaussian distributions[5] $\mathcal{N}(\mathbf{u}_{\boldsymbol{\theta}}, \Sigma_{\boldsymbol{\theta}})$ and $\mathcal{N}(\mathbf{u}_{\mathbf{w}}, \Sigma_{\mathbf{w}})$ are used to approximate the unknown posteriors of feature and sample weights, respectively. We start with the logarithm of Equation (9) by the following formula:

$$\begin{aligned} Q(\mathbf{w}, \boldsymbol{\theta}) &= \log\{p(\mathbf{t} \mid \mathbf{w}, \boldsymbol{\theta}, \mathrm{S}) p(\mathbf{w} \mid \boldsymbol{\alpha}) p(\boldsymbol{\theta} \mid \boldsymbol{\beta})\} - \log p(\mathbf{t} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathrm{S}) \\ &= \sum_{n=1}^{N} [t_n \log \sigma_n + (1 - t_n) \log(1 - \sigma_n)] - \frac{1}{2} \mathbf{w}^T \mathbf{A} \mathbf{w} - \frac{1}{2} \boldsymbol{\theta}^T \mathbf{B} \boldsymbol{\theta} \\ &\quad + \sum_{i=1}^{N} \log 1_{w_i \geq 0}(w_i) + \sum_{k=1}^{M} \log 1_{\theta_k \geq 0}(\theta_k) + \text{const}, \end{aligned}$$

where $\mathbf{A} = \text{diag}(\alpha_0, \ldots, \alpha_N)$, $\mathbf{B} = \text{diag}(\beta_1, \ldots, \beta_M)$, and const is independent of $\mathbf{w}$ and $\boldsymbol{\theta}$.

Using the sigmoid approximation, we substitute $1_{x \geq 0}(x)$ by $\sigma(\lambda x)$ with $\lambda = 5$. We can compute the derivative of the feature posterior function as follows:

$$\frac{\partial Q(\mathbf{w}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\mathbf{B}\boldsymbol{\theta} + \mathbf{D}^T(\mathbf{t} - \boldsymbol{\sigma}) + \mathbf{k}_{\boldsymbol{\theta}},$$

where $\mathbf{k}_{\boldsymbol{\theta}} = [\lambda(1 - \sigma(\lambda\theta_1)), \ldots, \lambda(1 - \sigma(\lambda\theta_M))]^T$ is an M-dimensional vector, $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_N]^T$, and $D = \frac{\partial \boldsymbol{\Phi}_{\boldsymbol{\theta}} \mathbf{w}}{\partial \boldsymbol{\theta}}$.

For Gaussian RBF,

$$D_{i,k} = -\sum_{j=1}^{N} w_j \phi_{\boldsymbol{\theta}, ij} \left(x_i^k - x_j^k\right)^2.$$

For Pth-order polynomial,

$$D_{i,k} = P x_i^k \sum_{j=1}^{N} w_j \phi_{\boldsymbol{\theta}, ij}^{(P-1)/P} x_j^k.$$

[5]Because of the truncated prior assumption, we should take the positive quadrant part of the two Gaussian distributions, which only have an extra normalization term. Fortunately, the normalization term is independent of $\mathbf{w}$ and $\boldsymbol{\theta}$. So in the derivation, we still use the Gaussian distributions.

The mean $\mathbf{u}_\theta$ of the feature posterior distribution is calculated by setting $\frac{\partial Q(\mathbf{w},\theta)}{\partial \theta} = 0$:

$$\mathbf{u}_\theta = \mathbf{B}^{-1}\left(\mathbf{D}^T(\mathbf{t}-\sigma) + \mathbf{k}_\theta\right). \tag{10}$$

Then, we compute the second-order derivative of $Q(\mathbf{w},\boldsymbol{\theta})$, the Hessian matrix:

$$\frac{\partial^2 Q(\mathbf{w},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2} = -\mathbf{O}_\theta - \mathbf{B} - \mathbf{D}^{\mathrm{T}}\mathbf{C}\mathbf{D} + \mathbf{E},$$

where $\mathbf{O}_\theta = \mathrm{diag}(\lambda^2\sigma(\lambda\theta_1)(1-\sigma(\lambda\theta_1)),\ldots,\lambda^2\sigma(\lambda\theta_M)(1-\sigma(\lambda\theta_M)))$ is an $M\times M$ diagonal matrix, and $\mathbf{C}$ is an $N\times N$ diagonal matrix $\mathbf{C} = \mathrm{diag}((1-\sigma_1)\sigma_1,\ldots,(1-\sigma_N)\sigma_N)$. $\mathbf{E}$ denotes $\frac{\partial \mathbf{D}}{\partial \theta}^T(\mathbf{t}-\sigma)$ and is computed as follows:

For Gaussian RBF,

$$E_{i,k} = \sum_{p=1}^{N}\left[(t_p-\sigma_p)\sum_{j=1}^{N}\phi_{\theta,pj}w_j\left(x_p^i - x_j^i\right)^2 \times \left(x_p^k - x_j^k\right)^2\right].$$

For Pth-order polynomial,

$$E_{i,k} = \sum_{p=1}^{N}\left[(t_p-\sigma_p)x_p^i x_p^k \sum_{j=1}^{N}\phi_{\theta,pj}^{(P-2)/P} w_j x_j^i x_j^k\right] \times P(P-1).$$

The covariance of this approximate posterior distribution equals the negative inverse of the Hessian matrix:

$$\Sigma_\theta = \left(\mathbf{D}^{\mathrm{T}}\mathbf{C}\mathbf{D} + \mathbf{B} + \mathbf{O}_\theta - \mathbf{E}\right)^{-1}. \tag{11}$$

Practically, we use Cholesky decomposition to compute the robust inversion.

In the same way, we can obtain $\mathbf{u}_\mathbf{w}$ and $\Sigma_\mathbf{w}$ by computing the derivative of $Q(\mathbf{w},\boldsymbol{\theta})$ with respect to $\mathbf{w}$:

$$\mathbf{u}_\mathbf{w} = \mathbf{A}^{-1}\left(\Phi_\theta^T(\mathbf{t}-\boldsymbol{\sigma}) + \mathbf{k}_\mathbf{w}\right), \tag{12}$$

$$\Sigma_{\boldsymbol{w}} = \left(\Phi_\theta^T\mathbf{C}\Phi_\theta + \mathbf{A} + \mathbf{O}_\mathbf{w}\right)^{-1}, \tag{13}$$

where $\mathbf{k}_\mathbf{w} = [0, \lambda(1-\sigma(\lambda w_1)),\ldots,\beta(1-\sigma(\lambda w_N))]^T$ is an $(N+1)$-dimension vector, and $\mathbf{O}_\mathbf{w} = \mathrm{diag}(0,\lambda^2\sigma(\lambda w_1)(1-\sigma(\lambda w_1)),\ldots,\lambda^2\sigma(\lambda w_N)(1-\sigma(\lambda w_N)))$ is an $(N+1)\times(N+1)$ diagonal matrix.

After the derivation, the indicator functions degenerate into vectors and matrices, $\mathbf{k}_\theta$ in Equation (10), $\mathbf{O}_\theta$ in Equation (11) for the feature posterior, $\mathbf{k}_\mathbf{w}$ in Equation (12), and $\mathbf{O}_\mathbf{w}$ in Equation (13) for the sample posterior. These two matrices will hold the non-negative property of the sample and feature weights, which is consistent with the prior assumption.

With the approximated posterior distributions, $\mathcal{N}(\mathbf{u}_\theta,\Sigma_\theta)$ and $\mathcal{N}(\mathbf{u}_\mathbf{w},\Sigma_\mathbf{w})$, optimizing PFCVM$_{\mathrm{LP}}$ boils down to maximizing the posterior mode of the hyperparameters, which means maximizing $p(\boldsymbol{\alpha},\boldsymbol{\beta}\mid\mathbf{t}) \propto p(\mathbf{t}\mid\boldsymbol{\alpha},\boldsymbol{\beta},\mathrm{S})p(\boldsymbol{\alpha})p(\boldsymbol{\beta})$ with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. As we use flat Gamma distributions over $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, the maximization depends on the marginal likelihood $p(\mathbf{t}\mid\boldsymbol{\alpha},\boldsymbol{\beta},\mathrm{S})$ [22, 45]. In the next section, the optimal marginal likelihood is obtained through the type-II maximum likelihood method.

3.2 Maximum Marginal Likelihood

In Bayesian models, the marginal likelihood function is computed as follows:

$$p(\mathbf{t} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathrm{S}) = \int p(\mathbf{t} \mid \mathbf{w}, \boldsymbol{\theta}, \mathrm{S}) p(\mathbf{w} \mid \boldsymbol{\alpha}) p(\boldsymbol{\theta} \mid \boldsymbol{\beta}) d\mathbf{w} d\boldsymbol{\theta}. \tag{14}$$

However, when the likelihood function is a Bernoulli distribution and the priors are approximated by Gaussian distributions, the maximization of Equation (14) cannot be derived in closed form. Thus, we introduce an iterative estimation solution. The details of the hyperparameter optimization and the derivation of maximizing marginal likelihood are specified in Appendix. Here, we use the methodology of Bayesian Occam's razor [32]. The update formula of the feature hyperparameters is rearranged and simplified as

$$\beta_k^{\text{new}} = \frac{\gamma_k}{u_{\theta,k}^2}, \tag{15}$$

where $u_{\theta,k}$ is the kth mean of feature weights in Equation (10), and we denote $\gamma_k \equiv 1 - \beta_k \Sigma_{kk}$, where Σ_{kk} is the kth diagonal covariance element in Equation (11), and $\beta_k \Sigma_{kk}$ works as Occam's factor, which can automatically find a balanced solution between complexity and accuracy of PFCVM_{LP}. The details of updating the sample hyperparameters α_i are the same as for β_k, and we omit them.

In the training step, we will eliminate a feature when the corresponding β_k is larger than a specified threshold. In this case, the feature weight θ_k is dominated by the prior distribution and restricted to a small neighborhood around 0. Hence, this feature contributes little to the classification performance. At the start of the iterative process, all samples and features are included in the model. As iterations proceed, N and M are quickly reduced, which accelerates the speed of the iterations. Further analysis of the complexity will be reported in Section 4.4. In the next subsection, we demonstrate how to make predictions on new data.

3.3 Making Predictions

When predicting the label of a new sample $\hat{x}$, instead of making a hard binary decision, we prefer to estimate the uncertainty in the decision, the posterior probability of the prediction $p(\hat{y} = 1 \mid \hat{x}, \mathrm{S})$. Incorporating the Bernoulli likelihood, the Bayesian model enables the sigmoid function $\sigma(f(\hat{x}))$ to be regarded as a consistent estimate of $p(\hat{y} = 1 \mid \hat{x}, \mathrm{S})$ [45]. We can compute the probability of prediction in the following way:

$$p(\hat{y} = 1 \mid \hat{x}, \mathrm{S}) = \int p(\hat{y} = 1 \mid \mathbf{w}, \hat{x}, \mathrm{S}) q(\mathbf{w}) d\mathbf{w},$$

where $p(\hat{y} = 1 \mid \mathbf{w}, \hat{x}, \mathrm{S}) = \sigma(\mathbf{u}_{\mathbf{w}}^T \boldsymbol{\phi}_{\boldsymbol{\theta}}(\hat{x}))$ and $q(\mathbf{w})$ denotes the posterior of sample weights. Employing the posterior approximation in Section 3.1, we have $q(\mathbf{w}) \approx \mathcal{N}(\mathbf{w} \mid \mathbf{u}_{\mathbf{w}}, \Sigma_{\mathbf{w}})$. According to [4], we have

$$p(\hat{y} = 1 \mid \hat{x}, \mathrm{S}) = \int \sigma\left(\boldsymbol{\phi}_{\boldsymbol{\theta}}^T(\hat{x}) \mathbf{u}_{\mathbf{w}}\right) N(\mathbf{w} \mid \mathbf{u}_{\mathbf{w}}, \Sigma_{\mathbf{w}}) d\mathbf{w} \approx \sigma\left(\kappa\left(\sigma_{\hat{x}}^2\right) \mathbf{u}_{\mathbf{w}}^T \phi_{\boldsymbol{\theta}}(\hat{x})\right),$$

where $\kappa(\sigma_{\hat{x}}^2) = (1 + \frac{\pi}{8} \boldsymbol{\phi}_{\boldsymbol{\theta}}^T(\hat{x}) \Sigma_{\mathbf{w}} \boldsymbol{\phi}_{\boldsymbol{\theta}}(\hat{x}))^{-1/2}$ is the variance of $\hat{x}$ with the covariance of sample posterior distribution $\Sigma_{\mathbf{w}}$.

To arrive at a binary classification, we choose $\mathbf{u}_{\mathbf{w}}^T \boldsymbol{\phi}_{\boldsymbol{\theta}}(\hat{x}) = 0$ as the decision boundary, where we have the probability $p(\hat{y} = 1 \mid \hat{x}, \mathrm{S}) = 0.5$. Thus, computing the sign of $\mathbf{u}_{\mathbf{w}}^T \boldsymbol{\phi}_{\boldsymbol{\theta}}(\hat{x})$ will meet the case of 0–1 classification. Moreover, the likelihood of prediction provides the confidence of the prediction, which is more important in unbalanced classification tasks.

3.4 Implementation

We detail the implementation of PFCVM$_{\text{LP}}$ step by step and provide pseudocode in Algorithm 1.

ALGORITHM 1: PFCVM$_{\text{LP}}$ algorithm

1: **Input:** Training data set: **S**; initial values: *INITVALUES*; threshold: *THRESHOLD*; the maximum number of iterations: *maxIts*.
2: **Output:** Weights of model: **WEIGHT**; Hyperparameters: **HYPERPARAMETER**.
3: **Initialization:** $[\mathbf{w}, \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}]$= *INITVALUES*; **Index** = generateIndex($\boldsymbol{\alpha}, \boldsymbol{\beta}$)
4: **while** $i <$ *maxItes* **do**
5: $\boldsymbol{\Phi}$ = *updateBasisFunction*($\mathbf{x}, \boldsymbol{\theta}$, **Index**)
6: $[\mathbf{w}, \boldsymbol{\theta}]$ = updatePosterior($\boldsymbol{\Phi}$, $\mathbf{w}, \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}$, Y)
7: $[\boldsymbol{\alpha}, \boldsymbol{\beta}]$ = maximumMarginal($\boldsymbol{\Phi}$, $\mathbf{w}, \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}$, Y)
8: **if** $\boldsymbol{\alpha}_i$ or $\boldsymbol{\beta}_k >$ *THRESHOLD.maximum* **then**
9: delete the ith sample or the kth feature
10: **end if**
11: **Index** = updateIndex($\boldsymbol{\alpha}, \boldsymbol{\beta}$)
12: *marginal* = calculateMarginal($\boldsymbol{\Phi}$, $\mathbf{w}, \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}$, Y)
13: **if** Δ*marginal* $<$ *THRESHOLD.minimal* **then**
14: break
15: **end if**
16: **WEIGHT** = $[\mathbf{w}, \boldsymbol{\theta}$, **Index**$]$
17: **HYPERPARAMETER** = $[\boldsymbol{\alpha}, \boldsymbol{\beta}]$
18: **end while**

Algorithm 1 consists of the following main steps:

(1) First, the values of $\mathbf{w}, \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}$ are initialized by *INITVALUES* and a parameter **Index** generated to indicate the useful samples and features (line 3).
(2) At the beginning of each iteration, compute the matrix $\boldsymbol{\Phi}$ according to Equation (3) (line 5).
(3) Based on Equation (9), use the new hyperparameters to re-estimate the posterior (line 6).
(4) Use the re-estimated parameters to maximize the logarithm of marginal likelihood and update the hyperparameters according to Equation (14) (line 7).
(5) Prune irrelevant samples and useless features if the corresponding hyperparameters are larger than a specified threshold (lines 8–10).
(6) Update the **Index** vector (line 11).
(7) Calculate the logarithm of the marginal likelihood (line 12).
(8) Convergence detection, if the change of marginal likelihood is relatively small, halt the iteration (lines 13–15).
(9) Generate the output values. The vector **WEIGHT** consists of sample and feature weights and the vector **Index** indicates the relevant samples and features (lines 16 and 17).

We have now presented all details of PFCVM$_{\text{LP}}$, including derivations of equations and pseudocode. Next, we evaluate the performance of PFCVM$_{\text{LP}}$ by comparing with other state-of-the-art algorithms on a Waveform (UCI) dataset, EEG emotion recognition datasets, and high-dimensional gene expression datasets.

4 EXPERIMENTAL RESULTS

In a series of experiments, we assess the performance of PFCVM_{LP}. The first experiment aims to evaluate the robustness and stability of PFCVM_{LP} against noise features. Second, a set of experiments are carried out on the emotional EEG datasets to assess the performance of classification and feature selection. Then, experiments are designed on gene expression datasets, which contain lots of irrelevant features. Finally, the computational and space complexity of PFCVM_{LP} is analyzed.

4.1 Waveform Dataset: Stability and Robustness Against Noise

The Waveform dataset [36] contains a number of noise features and has been used to estimate the robustness of feature selection algorithms. This dataset contains 5,000 samples with 3 classes of waves (about 33% for each wave). Each sample has 40 continuous features, in which the first 21 features are relevant for classification, whereas the latter 19 features are irrelevant noise with mean 0 and variance 1. The presence of 19 noise features in the Waveform dataset increases the hardness of the classification problem. Ideal feature selection algorithms should select the relevant features (features 1–21) and simultaneously remove the irrelevant noise features (features 22–40). To evaluate the stability and robustness of feature selection of PFCVM_{LP} with noise features, we choose wave 1 vs. wave 2 from the Waveform as the experimental data, which includes 3,345 samples. In the experiment, we randomly sample data examples to generate 100 distinct training and testing sets, in which each training set includes 200 training samples for each class. Then, we run PFCVM_{LP} and three embedded feature selection algorithms on each data partition.

First, to compare the stability of PFCVM_{LP} against that of other algorithms, two indicators are employed to measure the stability, i.e., the popular Jaccard index stability [23] and the recently proposed Pearson's correlation coefficient stability [40]. The stability in the output feature subsets is a key evaluation metric for feature selection algorithms, which quantizes the sensitivity of a feature selection procedure with different training sets. Assume $\mathcal{F}$ denotes the set of selected feature subsets, $\boldsymbol{s_i}$, $\boldsymbol{s_j} \in \mathcal{F}$ are two selected feature subsets. The *Jaccard index* between $\boldsymbol{s_i}$, $\boldsymbol{s_j}$ is defined as

$$\psi_{\text{Jaccard}}(\boldsymbol{s_i}, \boldsymbol{s_j}) = \frac{|\boldsymbol{s_i} \cap \boldsymbol{s_j}|}{|\boldsymbol{s_i} \cup \boldsymbol{s_j}|} = \frac{r_{ij}}{r_i + r_j - r_{ij}}, \tag{16}$$

where r_{ij} denotes the number of common features in $\boldsymbol{s_i}$ and $\boldsymbol{s_j}$, and r_i is the size of selected features in $\boldsymbol{s_i}$. Based on the Jaccard index in Equation (16), the *Jaccard stability* of $\mathcal{F}$ is computed as follows:

$$\Psi_{\text{Jaccard}}(\mathcal{F}) = \frac{2}{R(R-1)} \sum_{i=1}^{R-1} \sum_{j>i}^{R} \psi_{\text{Jaccard}}(\boldsymbol{s_i}, \boldsymbol{s_j}), \tag{17}$$

in which R denotes the number of the selected feature in $\mathcal{F}$. $\Psi_{\text{Jaccard}}(\mathcal{F}) \in [0, 1]$, where 0 means there is no overlap between any two feature subsets, 1 means that all feature subsets in $\mathcal{F}$ are identical.

Following [40], the Pearson's coefficient between $\boldsymbol{s_i}$ and $\boldsymbol{s_j}$ can be redefined as follows:

$$\psi_{\text{Pearson}}(\boldsymbol{s_i}, \boldsymbol{s_j}) == \frac{M \cdot r_{ij} - r_i \cdot r_j}{\sqrt{r_i \cdot r_j (M - r_i) \cdot (M - r_j)}}, \tag{18}$$

where M is the number of sample features. Using Equation (18), the Pearson's correlation coefficient stability value of $\mathcal{F}$ is computed as follows:

$$\Psi_{\text{Pearson}}(\mathcal{F}) = \frac{2}{R(R-1)} \sum_{i=1}^{R-1} \sum_{j>i}^{R} \psi_{\text{Pearson}}(\boldsymbol{s_i}, \boldsymbol{s_j}). \tag{19}$$

Table 1. The Jaccard and Pearson Stability Performances of PFCVM_{LP} and Other Embedded Feature Selection Algorithms on Waveform Dataset

Algorithms	PFCVM_{LP}	PFCVM_{EM}	WSVM	JCFO
Jaccard	**0.556**±0.071	0.525±0.082	0.543±0.076	0.518±0.072
Pearson	**0.662**±0.016	0.610±0.026	0.646±0.019	0.603±0.017

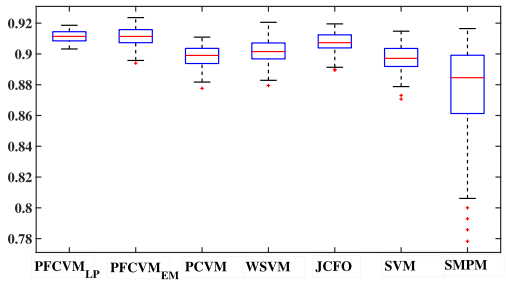


Fig. 2. Classification accuracy of PFCVM_{LP} and compared algorithms.

$\Psi_{\text{Pearson}}(\mathcal{F}) \in [-1, 1]$, in which -1 means that any two feature subsets are complementary, 0 means there is no correlation between any two feature subsets, and 1 means that all feature subsets in $\mathcal{F}$ are fully correlated.

In order to provide comprehensive results, three embedded feature algorithms, WSVM, JCFO, and PFCVM_{EM}, and three supervised learning algorithms, SVM, PCVM, and SMPM [18] using all features are chosen for comparison. The experimental settings are the same as those in [8]. The experiments are repeated 100 times with different training and test sets, and 100 feature subsets will be obtained. Therefore, the stability of each algorithm, measured by Jaccard index and Person's correlation coefficient, is listed in Table 1, and the classification accuracy is depicted in Figure 2.

According to the stability definition in Equations (17) and (19), a high value of stability means that the selected feature subsets do not significantly change with different training sets. From Table 1 and Figure 2, we observe that PFCVM_{LP} achieves the best stability performance in terms of both Jaccard and Pearson index, and highly competitive accuracy in comparison with other algorithms. The stability of WSVM is better than that of PFCVM_{EM} and JCFO, which is attributed to the use of the LIBSVM [7] and CVX [17] optimization toolbox. However, PFCVM_{EM} and JCFO show inferior stability scores, the reason being that they use the EM algorithm to a point estimate of feature parameters, which suffers from the initialization and may converge to a local optimum [9]. Finally, in Figure 2, we also note that due to the lack of feature selection, SVM and SMPM perform poorly.

In order to demonstrate the robustness of PFCVM_{LP} against the irrelevant noise features, the selected frequency of each feature, $\hat{P}_f$, is shown in Figure 3. From Figure 3, we observe that PFCVM_{LP} shows comparative effectiveness to WSVM, JCFO, and PFCVM_{EM} on the first 21 actual features, and the $\hat{P}_f$ of features 5, 9, 10, 11, 12, 15, 16, 17, 18 are greater than 0.5. As shown in Figure 2, using these features, SVM achieves 92.12% accuracy, an improvement over the result obtained by using all features. To quantitatively evaluate the capability of eliminating noise features for these embedded feature selection algorithms, the frequency of selecting the latter 19 noise features is used. From Figure 3, we note that the frequencies of selecting the noise features are all less than 0.2 in PFCVM_{LP}. However, there are 3, 1, 2 noise features with more than 0.2 selected frequencies for WSVM, JCFO, and PFCVM_{EM}, respectively. This result demonstrates that in the presence

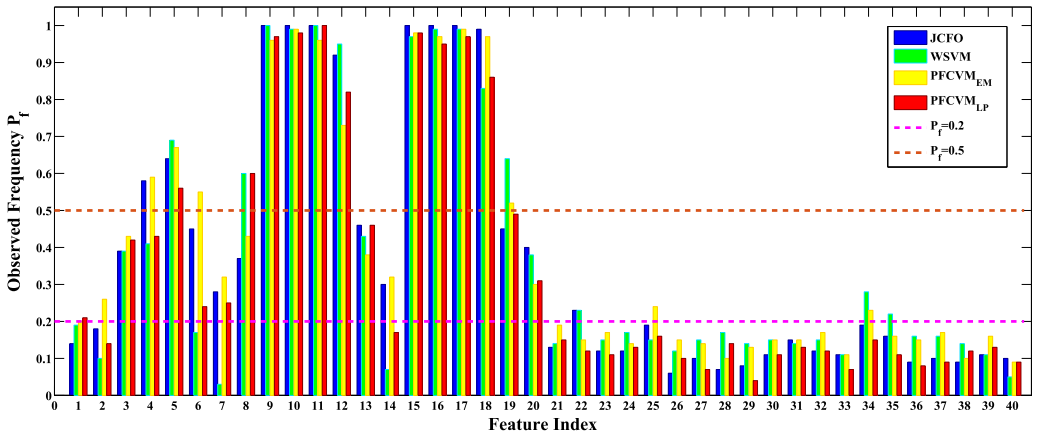


Fig. 3. The selected frequency of each feature. The first 21 features are actual features and the latter 19 features are noise.

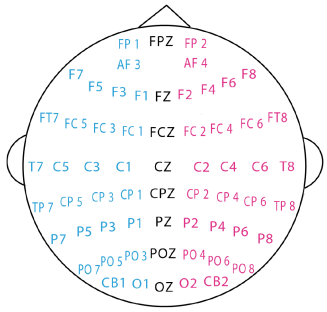


Fig. 4. The layout of 62-channel symmetrical electrodes on the EEG.

of noise features, PFCVM$_{LP}$ performs much better than other algorithms in terms of eliminating those noise features.

4.2 Emotional EEG Datasets: Emotion Recognition and Effectiveness for Feature Selection

In this section, a newly developed emotion EEG dataset, SEED [59], will be used to evaluate the performance of PFCVM$_{LP}$. The SEED dataset contains the EEG signals of 15 subjects, which were recorded while the subjects were watching 15 emotional film clips in the emotion experiment. The subjects' emotional reactions to the film clips are used as the emotional labels (−1 for negative, 0 for neutral, and +1 for positive) of the corresponding film clips. The EEG signals were recorded by 62-channel symmetrical electrodes which are shown in Figure 4. In our experiments, the differential entropy (DE) features are chosen for emotion recognition due to its better discrimination [13]. The DE features are extracted from five common frequency bands, namely Delta (1–3Hz), Theta (4–7Hz), Alpha (8–13Hz), Beta (14–30Hz), and Gamma (31–50Hz). Therefore, each frequency band has 62-channel symmetrical electrodes and there are totally 310 features for one sample. In order to investigate neural signatures and stable patterns across sessions and individuals, each subject performed the emotion experiment in three separate sessions with an interval of about one week or longer.

In this article, we choose *positive* vs. *negative* samples from SEED as our experimental data, which includes the signals of five positive and five negative film clips. In this experiment, the EEG signals recorded from one subject is regarded as one dataset, and thus there are 15 datasets for 15 subjects [60]. Each dataset has three sessions data, and each session contains 2,290 samples (1,120 negative samples and 1,170 positive samples) with 310 features. For each data, we choose 1,376 samples as the training set (recorded from three positive and three negative film clips), the remaining in the same session as a test set. We compare the emotion recognition and feature selection effectiveness of PFCVM_{LP} with other algorithms on the testing data.

To evaluate the emotion recognition performance, we take four supervised learning algorithms (i.e., SVM, SMPM, RVM, and PCVM) using all features as baselines and also compare PFCVM_{LP} with seven state-of-the-art feature selection algorithms: mRMR [41], TRC [39], FSNM [38], L_1SVM [5], WSVM [37], JCFO [26], and PFCVM_{EM} [29]. Among the algorithms considered, mRMR, TRC, and FSNM are filtered feature selection algorithms, L_1SVM, WSVM, JCFO, and PFCVM_{EM} are embedded feature selection algorithms. For mRMR, TRC, and FSNM, the PCVM classifier is used to evaluate their emotion recognition performance by using the features selected by them. In these experiments, the Gaussian RBF is used as the basis function. Two popular evaluation criteria, i.e., error rate[6] and area under the curve of the receiver operating characteristic (AUC) are adopted for evaluation; they represent a rank criterion and a threshold criterion, respectively [6].

We follow the procedure in [8] to choose the parameters. More precisely, the dataset for the 15th subject is chosen for cross-validation, in which we train each algorithm with all parameter candidates and then choose the parameters with the lowest median error rate on this dataset. We follow this procedure to choose the optimal numbers of clusters for SMPM, the kernel parameter ϑ for RVM, SMPM, PCVM, and WSVM, and the regularization parameters for JCFO and L_1SVM. For SVM, the regularization C and the kernel parameter ϑ are tuned by grid search, in which we train SVM with all combinations of each candidate C and ϑ, then choose the combination with the lowest median error rate. For mRMR, TRC, and FSNM, the proper sizes of feature subsets and the kernel parameter ϑ for PCVM are chosen by a similar grid search. We also choose the proper starting points for PFCVM_{EM} and the initial hyperparameters for PFCVM_{LP}.

For each subject, all algorithms are separately run on three session datasets, and the average results as well as the standard deviations of each algorithm are reported in Table 2. Furthermore, we conduct a statistical significance test by running the pair-wise t-test between PFCVM_{LP} and other algorithms, with 95% confidence level. Specifically, once PFCVM_{LP} achieves a significantly better/worse performance than others on a dataset, a win/loss is counted and we mark it with •/∘ in Table 2. Ties are also counted but are not marked. The number of wins/ties/losses of PFCVM_{LP} compared to other algorithms are shown in the last row of Table 2. In terms of the error rate, PFCVM_{LP} significantly outperforms other algorithms on most of datasets. But we note that PFCVM_{LP} is outperformed by L_1SVM, WSVM, and JCFO only on 1, 1, and 2 datasets, respectively. Specifically, on the subject #3 dataset, PFCVM_{LP} achieves 0% classification error, while the best competitor has a 3.24% error rate. In terms of the AUC, PFCVM_{LP} is only inferior to JCFO on the subject #8 dataset. These results indicate that PFCVM_{LP} outperforms the state-of-the-art feature selection algorithms on the EEG datasets.

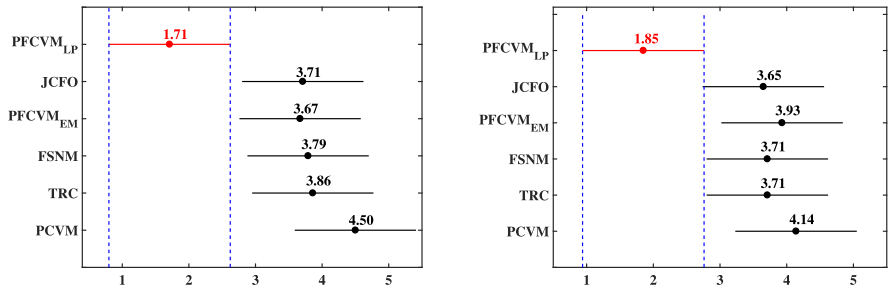
In order to give a comprehensive performance comparison between PFCVM_{LP} and other algorithms with statistical significance, the Friedman test [12] combining with the post-hoc tests is used to make comparisons of multiple methods over multiple datasets. The performance of two algorithms is significantly different if their average ranks on all datasets differ by at least the critical

[6]Error rate = 1 − classification accuracy = $\frac{1}{N}\sum_{i=1}^{N} 1(y_i \neq f(\mathbf{x}_i; \mathbf{w}, \boldsymbol{\theta}))$.

Table 2. The Error Rate and AUC of $PFCVM_{LP}$ and Other Algorithms on the Emotional EEG Datasets

Subject	The error rate (in %) of $PFCVM_{LP}$ and other algorithms											
	SVM	SMPM	RVM	PCVM	mRMR	TRC	FSNM	L_1SVM	WSVM	$PFCVM_{EM}$	JCFO	$PFCVM_{LP}$
#1	9.84±7.33●	8.35±7.46●	6.02±3.84●	8.72±3.02●	4.70±7.13	**4.52±5.48**	4.74±4.55	8.53±9.39●	5.80±5.04	6.85±4.57●	12.18±8.27●	4.63±4.02
#2	13.60±5.92	23.30±10.55●	12.78±8.72	13.42±5.18●	20.05±3.67●	22.43±13.15●	19.66±5.05●	18.09±9.26●	13.38±9.20	17.72±10.70●	19.11±8.33●	**11.85±4.74**
#3	5.03±6.13●	7.33±9.22●	8.28±10.34●	9.12±10.60●	9.70±6.80●	6.35±5.99●	3.14±5.43	8.77±8.93●	8.17±7.15●	4.96±4.41●	4.60±4.96●	**0.00±0.00**
#4	19.74±6.16●	20.45±6.19●	18.07±2.22●	19.77±1.86●	17.02±5.69	17.68±5.26●	16.65±6.23	17.28±6.43●	16.45±7.94	17.43±7.09●	19.31±9.65●	**16.12±5.15**
#5	18.23±7.57●	20.93±12.21●	19.82±9.20●	17.98±10.60●	19.93±7.72●	15.61±8.39	20.81±10.40●	18.05±10.08●	**13.57±9.86**	19.32±10.74●	20.81±11.11●	14.81±6.39
#6	12.31±6.04●	14.30±7.34●	9.09±9.36●	9.22±6.18●	7.95±6.65	10.44±9.45●	10.83±9.79●	16.45±8.25●	15.75±10.62●	8.01±4.95	11.77±4.99●	**6.16±5.24**
#7	10.69±9.15	10.20±10.78	11.34±9.83●	7.91±7.07	10.81±9.46●	9.70±8.72	14.26±11.99●	20.64±8.54●	13.13±11.51●	**7.48±6.60**	15.97±10.45●	8.04±7.10
#8	8.90±1.77●	5.93±1.26●	8.93±7.15●	7.35±4.74●	8.75±0.68●	4.47±2.52●	4.52±4.13●	3.00±2.65	5.29±5.15●	5.85±3.54●	**0.00±0.00**○	2.15±3.72
#9	14.85±9.42●	11.95±4.47	13.72±11.72●	12.61±11.72	13.15±8.54●	14.70±8.17●	10.07±10.74	8.28±8.61○	**7.70±6.69**○	16.30±12.68●	11.27±9.18	11.52±8.08
#10	14.33±8.11●	7.04±8.48●	12.50±10.32●	11.59±9.09●	8.49±8.41●	10.89±8.07●	9.23±8.27●	9.34±10.14●	16.74±10.59●	6.55±7.27●	5.27±1.04●	**3.06±2.98**
#11	12.21±5.63●	13.72±8.24●	12.65±10.16●	8.45±8.44●	7.09±7.47●	13.13±10.59●	13.57±9.96●	**1.46±2.53**	8.72±9.55●	1.93±1.97	2.04±2.66	1.98±2.29
#12	14.81±8.66●	12.74±7.35●	10.95±9.51●	12.69±3.34●	12.06±8.03●	10.69±2.94●	11.71±4.89●	6.08±3.77	9.32±5.81●	12.15±9.63●	**5.58±4.20**○	6.99±5.37
#13	10.83±4.86●	4.89±6.02	10.37±6.21●	7.17±6.65●	3.28±3.80	2.74±2.79	**2.12±2.25**	2.42±2.94	2.44±3.23	8.53±10.21●	4.08±2.43●	2.48±3.47
#14	15.46±1.12●	14.64±1.14●	15.40±2.89●	18.78±2.91●	17.79±3.78●	15.43±0.77●	19.69±2.58●	13.82±5.43	19.77±3.72●	**12.73±2.40**	13.62±8.04	13.27±3.51
Win/tie/loss	**12/2/0**	**11/3/0**	**13/1/0**	**12/2/0**	**11/3/0**	**10/4/0**	**9/5/0**	**9/5/1**	**8/5/1**	**9/5/0**	**8/4/2**	/
Subject	The AUC (in %) of $PFCVM_{LP}$ and other algorithms											
	SVM	SMPM	RVM	PCVM	mRMR	TRC	FSNM	L_1SVM	WSVM	$PFCVM_{EM}$	JCFO	$PFCVM_{LP}$
#1	98.37±1.62	97.47±1.98●	98.56±2.24	97.54±2.61●	97.64±2.09●	**99.70±0.46**	99.18±0.34	96.42±5.79●	98.04±3.39	96.29±3.22●	94.88±5.18●	99.54±0.50
#2	95.54±1.42●	86.79±6.82●	98.14±1.41	99.07±1.19	93.22±4.12●	92.46±8.08●	92.47±7.04●	94.98±5.24●	97.09±3.31●	94.01±5.88●	92.97±7.93●	**99.32±0.59**
#3	98.35±0.95●	97.75±1.32●	96.92±1.34●	99.84±0.28	94.15±4.12●	99.90±0.18	99.95±0.00	95.80±3.81●	94.43±2.72●	99.76±0.42	99.67±0.57	**100.00±0.00**
#4	90.27±6.77●	93.54±5.32	83.01±6.98●	82.18±4.50●	83.47±2.48●	93.55±3.58●	**95.20±5.74**	93.64±6.39●	91.53±7.88●	91.85±6.77●	94.88±4.87	95.05±4.67
#5	89.36±8.48●	82.42±10.21●	89.01±9.03●	93.62±9.25	92.87±9.39	87.02±9.85●	84.60±10.37●	84.41±10.88●	**93.91±6.25**	88.57±10.28●	84.76±10.38●	93.74±6.90
#6	95.33±1.85●	92.23±0.99●	94.15±2.97●	98.98±1.77	94.35±4.06●	95.11±1.64●	95.65±4.66●	93.87±7.54●	93.14±5.93●	97.39±2.55●	95.71±1.92●	**99.72±0.49**
#7	96.68±1.17●	95.23±3.37●	97.27±2.83●	99.23±0.86	99.12±0.92	99.17±1.29	95.72±4.95●	83.06±11.22●	96.19±3.68●	**99.57±0.42**	91.96±6.08●	99.17±0.76
#8	96.67±1.16●	97.05±0.75●	93.77±3.51●	95.94±4.74●	97.93±2.71	98.70±1.40	98.40±1.05	99.60±0.69	96.32±1.53●	97.33±3.89●	**100.00±0.00**○	98.72±1.96
#9	94.61±5.28●	93.10±5.26●	92.00±3.74●	94.90±4.26●	94.51±3.38●	95.30±4.67	95.20±5.31	95.46±7.87	**97.06±3.08**	94.40±5.47●	96.68±2.92	96.54±4.34
#10	93.70±4.42●	95.00±2.05●	95.04±4.94●	94.75±4.66●	95.59±4.49●	94.07±9.79●	97.91±2.14	96.67±5.76●	93.56±8.82●	94.05±9.21●	97.73±2.82	**98.08±1.67**
#11	92.48±7.22●	89.82±8.05●	88.68±10.40●	88.08±10.50●	91.23±8.35●	91.91±7.65●	92.61±8.86●	**100.00±0.00**	97.73±3.98●	99.28±1.15	98.74±0.65●	99.88±0.24
#12	87.45±9.90●	90.92±7.51●	96.21±3.23●	94.33±5.34●	95.69±7.30●	93.60±5.55●	92.38±8.08●	99.42±0.92	96.76±0.42●	95.36±4.26●	**99.87±0.23**	99.12±0.51
#13	98.88±1.98●	99.78±0.50	93.18±5.82●	97.84±3.73●	99.53±0.81	99.65±0.60	**100.00±0.00**	99.80±0.35	99.85±0.26	96.31±3.01●	99.33±1.16	99.91±0.16
#14	95.55±1.51	95.29±2.21	95.05±3.04	93.44±1.51●	94.15±3.97●	95.62±1.00	93.69±2.51●	95.94±3.71	91.28±1.23●	95.45±2.48	95.19±3.64	**96.15±3.45**
Win/tie/loss	**12/2/0**	**11/3/0**	**11/3/0**	**9/5/0**	**10/4/0**	**7/7/0**	**7/7/0**	**8/6/0**	**10/4/0**	**10/4/0**	**6/7/1**	/

The best result on each dataset is bolded. The symbols ●/○ indicate that $PFCVM_{LP}$ is significantly superior/inferior to other algorithms according to the pair-wise *t*-test with 95% confidence level.



(a) Average ranks of error rate (b) Average ranks of AUC

Fig. 5. Results of the Friedman test for the performance of PFCVM_{LP} and other algorithms on the EEG datasets. The dots denote the average ranks, the bars indicate the critical differences CD, and the algorithms having non-overlapped bars are significantly inferior to PFCVM_{LP}.

difference (CD):

$$\text{CD} = q_\alpha \sqrt{\frac{p(p+1)}{6N}}, \tag{20}$$

where p is the number of algorithms, N is the number of datasets, α is the significance level, and q_α denotes the critical value. According to the experimental results in Table 2, the better embedded feature selection algorithms, JCFO and PFCVM_{EM}, the better filter feature selection algorithms, TRC and FSNM, and the important baseline, PCVM, are chosen to compare with PFCVM_{LP}. Choosing $\alpha = 0.05$ and $q_\alpha = 2.576$ ($p = 6$), the critical difference becomes CD = 1.82.

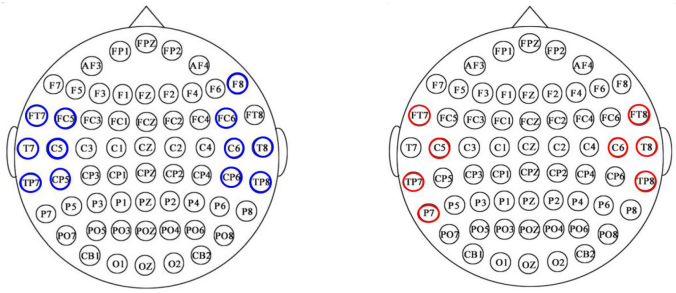
Figure 5 shows the Friedman test results on the EEG datasets. We observe that the differences between PFCVM_{LP} and PCVM are significant (greater than CD), demonstrating the effectiveness of simultaneously learning feature weights in terms of improving comprehensive performance. We note that the differences between PFCVM_{LP} and PFCVM_{EM} are greater than CD, which indicates that the Bayesian estimation approximated by the type-II maximum likelihood approach works better than the MAP point estimation based on the EM algorithm. Moreover, PFCVM_{LP} achieves a significant difference compared with other feature selection algorithms (i.e., TRC, FSNM, and JCFO), which demonstrates the effectiveness and superiority of PFCVM_{LP} for selecting relevant features.

To quantitatively assess the reliability of our classification results, the kappa statistic [47] is adopted to evaluate the consistency between the prediction of algorithms and the truth. The kappa statistic can be used to measure the performance of classifiers, which is more robust than classification accuracy. In this experiment, the kappa statistic of all classifiers ranges from 0 to 1, where 0 indicates the chance agreement between the prediction and the truth, and 1 represents a perfect agreement between them. Therefore, a larger kappa statistic value means that the corresponding classifier performs better. Table 3 reports the kappa statistic for PFCVM_{LP} and other algorithms on the emotional EEG datasets. The standard error interval with two-sided 95% confidence level of PFCVM_{LP} is also reported in Table 3, which constitutes the confidence interval of kappa statistic. The kappa statistics of other algorithms lying in the confidence interval of PFCVM_{LP} are marked with *, which indicates that they are not significantly worse or better than PFCVM_{LP}. From Table 3, we observe that PFCVM_{LP} achieves the best kappa statistics on 5 out of 14 datasets. Although other algorithms achieve the best kappa statistics on the rest datasets, they are not significantly better than PFCVM_{LP} since the best kappa statistics lie within the confidence interval of PFCVM_{LP} except on the subject 8 dataset. Overall, PFCVM_{LP} is the most frequent winner in terms of kappa statistic.

Table 3. The Kappa Statistic (in %) of $PFCVM_{LP}$ and Other Competing Algorithms on the Emotional EEG Datasets

Subject	SMPM	PCVM	mRMR	TRC	FSNM	L_1-SVM	WSVM	$PFCVM_{EM}$	JCFO	$PFCVM_{LP}$
#1	83.38	82.51	90.20*	**90.93***	90.45*	82.92	89.75*	89.28*	74.82	90.35 ± 2.84
#2	48.75	72.80	47.18	54.08	59.98	63.15	74.47	71.97	68.25	**78.55** ± 3.79
#3	85.06	81.34	81.26	87.09	93.67	85.96	87.08	92.56	93.28	**100.00** ± 0.00
#4	62.49	62.90	46.96	45.07	78.79*	73.65	74.47	74.37	55.80	**78.88** ± 3.12
#5	52.91	63.62	68.51	68.40	51.84	63.42	**79.76***	74.76	66.73	78.78 ± 3.30
#6	66.55	84.58	**90.63***	74.70	78.36	66.71	78.43	84.09	66.47	89.60 ± 2.63
#7	60.20	**84.25***	78.65	80.75	71.86	58.94	80.79*	83.97*	80.25	83.99 ± 3.51
#8	90.10	82.13	78.41	92.85	92.77	94.25	92.02	86.39	**100.00**	96.90 ± 1.36
#9	75.18	75.22	74.23	70.34	79.82	86.43*	**87.60***	75.97	83.91*	85.01 ± 2.69
#10	85.98	76.23	82.56	76.16	81.53	81.27	74.97	82.63	87.04*	**90.32** ± 3.53
#11	92.53	82.76	67.81	73.89	72.90	**95.09***	86.95	93.86*	93.29*	94.11 ± 2.69
#12	90.40*	74.47	77.59	78.57	76.41	91.78	86.42	80.16	**92.50***	91.97 ± 1.77
#13	90.14	73.72	96.05*	94.51	95.75*	95.75*	96.39*	93.06	94.50	**96.73** ± 1.33
#14	**78.78***	60.88	63.01	69.32	60.38	72.52*	70.38	77.12*	74.40*	75.02 ± 4.14

The best result for each dataset is illustrated in boldface.



(a) Beta band (b) Gamma band

Fig. 6. Profiles of top 20 features selected by $PFCVM_{LP}$ on the Beta and Gamma frequency bands.

According to Zheng and Lu [59], there are several irrelevant EEG channels in the SEED dataset, which will introduce noise to emotion recognition, and degrade the performance of classifiers. To illustrate the ability of $PFCVM_{LP}$ for selecting discriminative features and remove irrelevant features, Figure 6 illustrates the positions of the top 20 features selected by $PFCVM_{LP}$. As depicted in the figure, the top 20 features are all from the Beta and Gamma frequency bands and located at the lateral temporal area, which is consistent with previous findings [59, 60]. This result indicates that $PFCVM_{LP}$ can effectively select the relevant channels containing discriminative information and simultaneously eliminate irrelevant channels for emotion recognition task.

4.3 High-Dimensional Gene Expression Data: Performance in the Presence of Many Irrelevant Features

Gene expression datasets contain lots of irrelevant features, which may degrade the performance of classifiers [1, 16, 25]. In this experiment, three gene expression datasets: colon cancer [1], Duke cancer [50], and ALLAML [16] are chosen to examine whether $PFCVM_{LP}$ is able to eliminate the irrelevant features and make informative predictions for high-dimensional data. The colon cancer dataset includes expression levels of 2,000 gene features from 62 different samples, in which

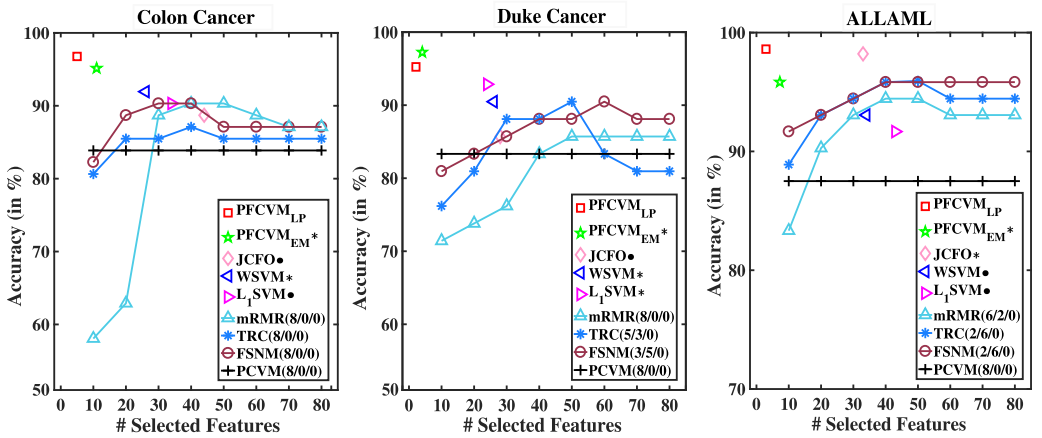


Fig. 7. Accuracy curves of different algorithms with different numbers of selected features. The symbols •/∗ denote win/tie of PFCVM_{LP} comparing to the other algorithms, and the three numbers in brackets represent the times of win/tie/loss of PFCVM_{LP} comparing to the corresponding algorithms by the binomial test at the 5% significance level.

40 samples are tumor colon and 22 normal colon tissues. The Duke cancer dataset contains expression levels of 7,129 genes from 42 tumor samples, in which 21 samples are estrogen receptor-positive tumors and the rest of the samples are estrogen receptor-negative tumors. The ALLAML dataset comes from 47 normal and 40 cancer tissues. The ALLAML dataset consists of 72 samples in two classes, acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML), which have 47 and 25 instances, respectively. Each sample is represented by 7,129 gene expression values.

Considering the relatively small numbers of samples vs. the large numbers of features, in this set of experiments, we use the leave one out cross-validation method to compute the accuracy: we train on $n-1$ samples first and then test the classification model on the remaining sample. So we generate 62, 42, and 72 runs for the colon cancer dataset, the Duke cancer dataset, and the ALLAML dataset, respectively. Following [43], in preprocessing each dataset is normalized in two ways: sample-wise to follow a standard normal distribution and then dimension-wise to follow a standard normal distribution.

We compare PFCVM_{LP} with PCVM and three filter feature selection algorithms, including mRMR, TRC, and FSNM, with PCVM as the classifier. As before, we also compare PFCVM_{LP} with other feature and classifier co-learning algorithms, i.e., L_1SVM, WSVM, JCFO, and PFCVM_{EM}. As the dimensions of gene data are relatively high, we choose the inner product, i.e., the linear kernel, as the metric. Finally, we report the average accuracies of 62, 42, and 72 runs on the Colon cancer, Duke cancer, and ALLAML datasets, respectively. To get the statistical significance results, we run the binomial test between PFCVM_{LP} and other algorithms at the 5% significance level.

Figure 7 shows the accuracy curves of feature selection algorithms. For the filter-based feature selection algorithms, i.e., mRMR, TRC, and FSNM, we first select a subset of features in the range of $[10, 20, \ldots, 80]$. And then we train a PCVM with each number of selected features to obtain the accuracy. The symbols •/∗ denote the wins/ties of PFCVM_{LP} against the embedded methods, and the numbers in brackets represent the win/tie/loss counts for PFCVM_{LP} against PCVM and the filter methods with different numbers of selected features. From Figure 7, we see that PFCVM_{LP} selects the smallest subsets of features on all datasets, and achieves the highest accuracies on two out of three datasets. However, we note that the gaps between PFCVM_{LP} and competing algorithms are small. The binomial test indicates that all filter-based feature selection algorithms are

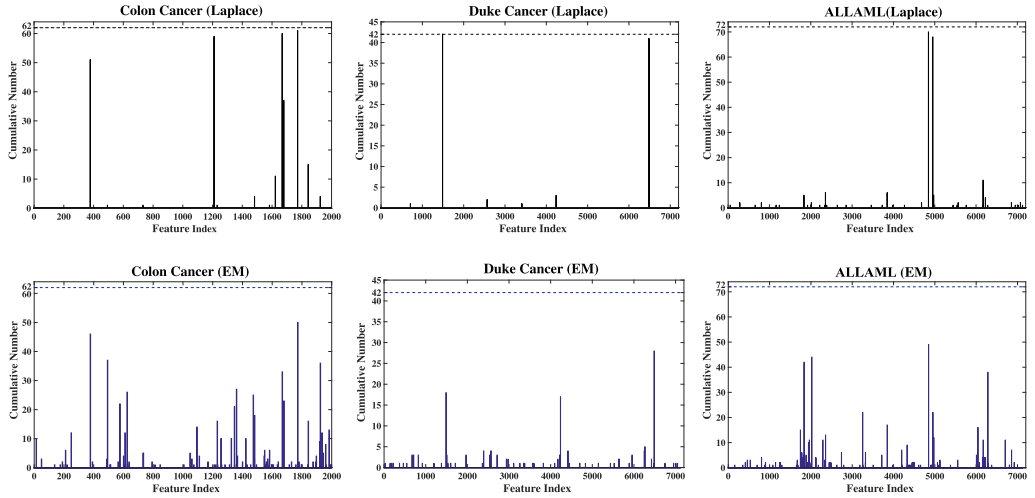


Fig. 8. Illustrations of selected features on gene expression datasets. The horizontal axis shows the index of features and the vertical axis shows the cumulative number of occurrences for the corresponding feature. The dashed line in each figure indicates the maximum cumulative number. The figures at the top show the features selected by PFCVM_{LP}; those at the bottom show the features selected by PFCVM_{EM}.

outperformed by PFCVM_{LP} on the Colon cancer dataset. And they can only reach the same level of performance as PFCVM_{LP} by including much more features on the other two datasets. Except for PFCVM_{EM}, PFCVM_{LP} also outperforms other embedded feature selection algorithms, i.e., JCFO, WSVM, and L_1SVM on some of datasets.

Figure 7 shows that the performances of PFCVM_{EM} are close to those of PFCVM_{LP}. But in our experiments, we find that the features selected by PFCVM_{EM} are more noisy. We show the cumulative number of occurrences[7] for each feature on gene expression datasets with PFCVM_{LP} and PFCVM_{EM}, in Figure 8. The numbers of features selected by two algorithms are similar on average, but from Figure 8, we see that PFCVM_{LP} concentrates on smaller sets of relevant features than PFCVM_{EM}. This is because that the Bayesian solution approximated by the type-II maximum likelihood is more stable than a solution achieved by the EM algorithm. This result and the results shown in Figure 7 together demonstrate that PFCVM_{LP} outperforms the state-of-the-art on the gene expression datasets in terms of accuracy.

In terms of feature selection effectiveness, PFCVM_{LP} averagely selects 4.94 features, on the colon cancer dataset. Among all 2,000 genes, five of them are particularly important (occurring in more than half of the tests). The biological explanations of these five genes are reported in Table 4 and two of them (No. 377 and No. 1772) are the same genes selected by Li et al. [31]. On the Duke cancer dataset, PFCVM_{LP} on average selects 2.14 features and two of them are selected in almost every run. On the ALLAML dataset, PFCVM_{LP} selects 2.94 features on average, and five of the seven most occurred genes are among the 50 genes most correlated with the diagnosis [16]. The biological significance of these genes is reported in Table 5.

To analyze the usefulness of the selected feature subsets by PFCVM_{LP} for other algorithms, we run $\text{RVM}_{\text{PFCVM}_{\text{LP}}}$, $\text{SVM}_{\text{PFCVM}_{\text{LP}}}$, and $\text{SMPM}_{\text{PFCVM}_{\text{LP}}}$ with features selected by PFCVM_{LP} and compare them with the original RVM, SVM, and SMPM. The results are reported in Table 6. According to

[7]We use the phrase "cumulative number of occurrences" to refer to the number of times a feature is selected in the experiments, e.g., if a feature is never selected in the experiments, its "cumulative number" is 0.

Table 4. Biological Significance of the Most Frequently Occurring Genes in the Colon Cancer Dataset

Feature ID	#	GenBank ID	Description [1]
1772	61	0H8393	Collagen alpha 2(XI) chain
1668	60	M82919	mRNA for GABAA receptor
1210	58	R55310	Mitochondrial processing peptidase
377	51	R39681	Eukaryotic initiation factor
1679	37	X53586	mRNA for integrin alpha 6

# denotes the number of occurrences for a feature in all runs.

Table 5. Biological Significance of the Most Frequently Occurring Genes in the ALLAML Cancer Dataset

Feature ID	#	GenBank ID	Relation	Description [16]
4847*	70	X95735	AML	Zyxin
4951	68	Y07604	N/A	Nucleoside diphosphate
6169	11	M13690	AML	Hereditary angioedema
3847*	6	U82759	AML	HoxA9 mRNA
2354*	6	M92287	AML	CCND3, Cyclin D3
4973*	5	Y08612	ALL	Protein RABAPTIN-5
1834*	5	M23197	AML	CD33 antigen

# denotes the number of occurrences for a feature in all runs. The superscript * denotes that these genes are among the top 50 most important genes for diagnosing AML/ALL [16].

Table 6. Accuracy on the Gene Expression Datasets

Accuracy (in %)	RVM	SVM	SMPM	$\text{RVM}_{\text{PFCVM}_{\text{LP}}}$	$\text{SVM}_{\text{PFCVM}_{\text{LP}}}$	$\text{SMPM}_{\text{PFCVM}_{\text{LP}}}$	PFCVM_{LP}
Colon cancer	85.48●	83.87●	75.81●	87.10●	85.48●	86.26●	**96.77**
Duke cancer	80.95●	85.71●	80.95●	92.86∗	**97.62∗**	90.48●	95.24
ALLAML	93.06●	87.50●	76.39●	95.83∗	95.83∗	94.44∗	**98.61**
Win/tie/loss	**3/0/0**	**3/0/0**	**3/0/0**	**1/2/0**	**1/2/0**	**2/1/0**	/

The symbols ●/∗ represent the win/tie of PFCVM_{LP} against others algorithms according to the binomial test at the 5% significance level.

this table, using the selected features by PFCVM_{LP}, $\text{RVM}_{\text{PFCVM}_{\text{LP}}}$, $\text{SVM}_{\text{PFCVM}_{\text{LP}}}$, and $\text{SMPM}_{\text{PFCVM}_{\text{LP}}}$ achieve better performances comparing to the original methods. Even to our surprise, $\text{SVM}_{\text{PFCVM}_{\text{LP}}}$ outperforms PFCVM_{LP} and obtains the best prediction on the Duke cancer dataset. This improvement demonstrates that the feature subsets selected by PFCVM_{LP} also work well for other algorithms.

4.4 Complexity Analysis

While computing the posterior covariance Σ_θ in Equation (11), we have to derive the negative inverse of the Hessian matrix. This derivation does not guarantee a numerically accurate result, because of the ill-condition of this Hessian matrix. Practically, we abandon the term $\mathbf{E}$ in Equation (11), so that the calculation becomes

$$\Sigma_\theta = \left(\mathbf{D}^{\mathbf{T}}\mathbf{C}\mathbf{D} + \mathbf{B} + \mathbf{O}_\theta\right)^{-1}. \tag{21}$$

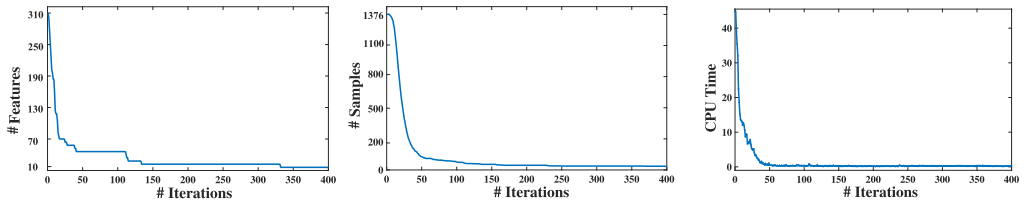


Fig. 9. Illustration of the rapid decrease in the size of features, samples, and CPU time. In these experiments, we choose the subject 1 dataset as an example.

In Equation (21), $\mathbf{B}$ and $\mathbf{O}_{\boldsymbol{\theta}}$ are positive definite diagonal matrices and $\mathbf{D}^{\mathbf{T}}\mathbf{C}\mathbf{D}$ has a quadratic form. Theoretically, the Hessian is a positive definite matrix. Nevertheless, because of machine precision, ill condition may still occur occasionally, especially when β_k is very large.

In the case of large β_k, especially when $\beta_k \to \infty$, the corresponding feature weight θ_k is restricted to a small neighborhood around 0. So during the iteration, we filter out this feature from our model. Initially, all the features are contained in the model. The main computational cost is the Cholesky decomposition in computing covariances of posteriors, $\Sigma_{\boldsymbol{\theta}}$ and $\Sigma_{\mathbf{w}}$, which is $O(N^3 + M^3)$. Thus, the computational complexity of PFCVM_{LP} is the same as that of PFCVM_{EM}, RSFM [35], and JCFO [26].

As for the storage requirements for PFCVM_{LP}, the basis function matrix $\boldsymbol{\Phi}_{\boldsymbol{\theta}}$ needs $O(N^2)$ for storage, and in the initial training stage the covariance matrices $\Sigma_{\boldsymbol{\theta}}$ and $\Sigma_{\mathbf{w}}$ require $O(M^2)$ and $O(N^2)$ storage, respectively. Therefore, the overall space complexity of PFCVM_{LP} is $O(N^2 + M^2)$, which is better than RSFM with a $O(N^2M + M^2)$ space complexity. As iterations proceed, N and M are rapidly decreasing, resulting in $O(\bar{N}^3 + \bar{M}^3)$ computational complexity and $O(\bar{N}^2 + \bar{M}^2)$ space complexity, where $\bar{N} \ll N$ and $\bar{M} \ll M$. In our experiments, N and M rapidly decrease to relatively small numbers in the first few iterations and the training speed quickly accelerates.

As illustrated in Figure 9, during the first 40 iterations the size of features and samples decreases from 310 to 40 and from 1,376 to 127, respectively, and the CPU time for each iteration step is decreased to 2.7% of the first iteration.

5 GENERALIZATION AND SPARSITY

Both the emotional EEG and gene expression experiments indicate that the proposed classifier and feature selection co-learning algorithm is capable of generating a sparse solution. In this section, we first analyze the KL-divergence between the prior and posterior. Following this, we investigate the entropic constraint Rademacher complexity [33] and derive a generalization bound for PFCVM_{LP}. By tightening the bound, we theoretically demonstrate the significance of the sparsity assumption and introduce a method to choose the initial values for PFCVM_{LP}.

5.1 KL-Divergence Between Prior and Posterior

In Bayesian learning, we use KL-divergence to measure the information gain from prior to posterior. As discussed in Section 3.1, the approximated posterior over feature parameters, denoted as $\tilde{q}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} \mid \mathbf{u}_{\boldsymbol{\theta}}, \Sigma_{\boldsymbol{\theta}})$, is a multivariate Gaussian distribution. However, as the feature prior is the left-truncated Gaussian prior, the true posterior over feature parameters should be restricted to the positive quadrant. In order to achieve this, we first compute the probability mass of the posterior in this half, $Z_0 = \int_0^{\infty} \tilde{q}(\boldsymbol{\theta}) d\boldsymbol{\theta}$; after that, we obtain a re-normalized version of the posterior: $q(\boldsymbol{\theta}) = \tilde{q}(\boldsymbol{\theta})/Z_0$, where $\theta_k \geq 0$.

We denote $\boldsymbol{\beta}_0 = (\beta_{0,1}, \beta_{0,2}, \ldots, \beta_{0,M})$ as the initial prior and $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_M)$ as the optimized prior. Following [9], we adopt the independent posterior assumption. We compute

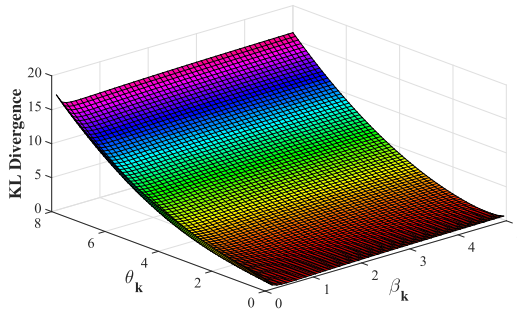


Fig. 10. Illustration of the numerical contribution of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ to $KL_{\boldsymbol{\theta}}(q\|p)$. To evaluate this contribution alone, we assume each sample weight $w_i = 1$ and each sample hyperparameter $\alpha_i = 1$.

$KL_{\boldsymbol{\theta}}(q\|p)$[8] using the following formula (the details are specified by [10]):

$$KL_{\boldsymbol{\theta}}(q\|p) = \int_0^{\infty} q(\boldsymbol{\theta}) \frac{\ln q(\boldsymbol{\theta})}{\ln p(\boldsymbol{\theta} \mid \boldsymbol{\beta}_0)} d\boldsymbol{\theta} = \sum_{k,\theta_k \neq 0} \left\{ \begin{array}{l} \frac{1}{2}\left[\frac{\beta_{0,k}}{\beta_k} - 1 + \ln\left(\frac{\beta_k}{\beta_{0,k}}\right) + \beta_{0,k}\theta_k^2\right] \\ + \frac{(2\pi\beta_k)^{-1/2}\left(\beta_{0,k}+\beta_k\right)\theta_k}{\mathrm{erfcx}\left(-\theta_i\sqrt{\beta_k/2}\right)} \\ - \ln\left(\mathrm{erfc}\left(-\frac{\theta_k\beta_k}{2}\right)\right) \end{array} \right\},$$

where $\mathrm{erfc}(z) = \frac{2}{\sqrt{\pi}} \int_z^{\infty} e^{-t^2} dt$ and $\mathrm{erfcx}(z) = e^{z^2}\mathrm{erfc}(z)$.

Note that $Z_{0,k} = \int_0^{\infty} \tilde{q}(\theta_k) d\theta_k = 1/2\,\mathrm{erfc}(-\theta_k\sqrt{\beta_k/2})$, so we can calculate $KL_{\boldsymbol{\theta}}(q\|p)$ as

$$KL_{\boldsymbol{\theta}}(q\|p) = \sum_{k,\theta_k \neq 0} \left\{ \begin{array}{l} \frac{1}{2}\left[\frac{\beta_{0,k}}{\beta_k} - 1 + \ln\left(\frac{\beta_k}{\beta_{0,k}}\right) + \beta_{0,k}\theta_k^2\right] \\ + \frac{(2\pi\beta_k)^{-1/2}\left(\beta_{0,k}+\beta_k\right)\theta_k}{2\exp\left(\beta_k\theta_k^2/2\right)} Z_{0,k}^{-1} \\ - \ln\left(Z_{0,k}\right) + \ln\left(\frac{\mathrm{erfc}\left(-\theta_k\sqrt{\beta_k/2}\right)}{2\,\mathrm{erfc}(-\theta_k\beta_k/2)}\right) \end{array} \right\}.$$

The KL-divergence is dominated by two parameters: $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$. However, the sensitivity of $KL_{\boldsymbol{\theta}}(q\|p)$ to these two parameters is different. As shown in Figure 10, setting the initial hyperparameter $\beta_{0,k} = 0.5$, we see that when changing the value of $\boldsymbol{\theta}$, the curve of $KL_{\boldsymbol{\theta}}(q\|p)$ shows significant changes, while this curve changes little when changing the optimized $\boldsymbol{\beta}_k$. Also, the minimum of $KL_{\boldsymbol{\theta}}(q\|p)$ is near the $\theta_k = 0$, where the corresponding feature is pruned.

5.2 Rademacher Complexity Bound

For a binary classification learning problem, the goal is to learn a function $f : \mathbb{R}^M \to \{-1, +1\}$ from a hypothesis class F, with the given dataset $\mathrm{S} = \{x_i, y_i\}_{i=1}^N$ drawn i.i.d. from a distribution D. We attempt to assess f by the expectation loss: $L(f) = E_{(x,y)\sim D} l(y, f(x))$, where $l(y, f(x))$ is a loss function. Practically, D is unaccessible and we can only assess the empirical loss for the given dataset S: $\Lambda(f, \mathrm{S}) = \frac{1}{N}\sum_{i=1}^N l(y_i, f(x_i))$. We adopt a 0–1 loss function: $l_{0\text{-}1}(y, f(x)) = I(yf(x) \geq 0)$, where $I(\cdot)$ is the indicator function. The loss function is dominated by the $1/c$-Lipschitz function: $l_c(a) = \min(1, \max(0, 1 - a/c))$, namely $l_{0\text{-}1}(y, f(x)) \leq l_c(yf(x))$. Then, we conclude the entropic constraint Rademacher complexity bound in the following theorem:

[8] $KL_{\boldsymbol{\theta}}(q\|p)$ denotes the KL-divergence between the posterior and prior in feature weights; p and q are short for $p(\boldsymbol{\theta} \mid \boldsymbol{\beta})$ and $q(\boldsymbol{\theta})$, respectively.

Theorem 1 ([9, 33]). *Based on the posterior $q(\mathbf{w}, \boldsymbol{\theta})$ given in Section 3.1, we have the Bayesian voting classifier:*

$$\hat{y} = f(x, q) = E_{q(\mathbf{w},\theta)}[\text{sign}(\Phi_{\boldsymbol{\theta}}(\mathbf{x})\mathbf{w})]. \tag{22}$$

Define $r > 0$ and $g > 0$ as arbitrary parameters. For all $f \in F$, defined at the start of Section 5.2, with probability at least $1 - \delta$, the bound for the generalization error of PFCVM$_{\text{LP}}$ on a given dataset S holds:

$$P(yf(x) < 0) \leq \Lambda(f, \mathbf{S}) + \frac{2}{c}\sqrt{\frac{2\tilde{g}(q(\mathbf{w}, \boldsymbol{\theta}))}{N}} + \sqrt{\frac{\ln \log_r \frac{r\tilde{g}(q(\mathbf{w},\boldsymbol{\theta}))}{g} + \frac{1}{2}\ln\frac{1}{\delta}}{N}}, \tag{23}$$

where c is the $1/c$-Lipschitz parameter, the empirical loss $\Lambda(f, \mathbf{S}) = \frac{1}{N}\sum_{i=1}^{N} l_c(y_i f(x_i))$, and the Rademacher entropic constraints $\tilde{g}(q(\mathbf{w}, \boldsymbol{\theta})) = r \cdot \max\{KL(Q\|P), g\}$. $KL(Q\|P)$ is the KL-divergence between the posteriors and priors in the sample and feature weights.

According to Equation (23), we observe that with a constant training set, the generalization error of PFCVM$_{\text{LP}}$ is mainly bounded by the empirical loss and $\tilde{g}(q(\mathbf{w}, \boldsymbol{\theta}))$, in which the latter is determined by $KL(Q\|P)$. Therefore, when the empirical loss is acceptable, a smaller $KL(Q\|P)$ could lead to a tighter bound. The contribution of $\mathbf{w}$ (the sample weight) has been analyzed in [9]. In order to analyze the effect of $\boldsymbol{\theta}$ (the feature weight) alone, we can assume the $\mathbf{w}$ is a given constant value $\hat{\boldsymbol{w}}$. As a result, we have $q(\hat{\boldsymbol{w}}, \boldsymbol{\theta}) = q(\boldsymbol{\theta} \mid \hat{\mathbf{w}})$ and thus $KL(Q\|P) = KL_{\boldsymbol{\theta}|\hat{\boldsymbol{w}}}(q\|p)$. As shown in Figure 10, the minimal $KL_{\boldsymbol{\theta}|\hat{\boldsymbol{w}}}(q\|p)$ is near $\theta_k = 0$. This is consistent with our prior assumption and demonstrates that a truncated Gaussian (sparse) prior over features can benefit the generalization performance by running as a regularization term and simultaneously encourage sparsity in feature space. Furthermore, in our model, the posterior and marginal likelihood are maximized iteratively in the training step. To accelerate the speed of convergence, we may choose proper starting points by minimizing $KL_{\boldsymbol{\theta}|\mathbf{w}}(q\|p)$, i.e., as indicated at the end of Section 5.1, we can use an optimal $\boldsymbol{\beta}$ instead of $\boldsymbol{\beta}_0$ as initial hyperparameter.

6 CONCLUSION

We have proposed a joint classification and feature learning algorithm PFCVM$_{\text{LP}}$. The proposed algorithm adopts sparseness-promoting priors for both sample and feature weights to jointly learn to select the informative samples and features. By using the Laplace approximation, we compute a complete Bayesian estimation of PFCVM$_{\text{LP}}$, which is more stable than previously considered EM-based solutions. The performance of PFCVM$_{\text{LP}}$ has been examined according to two criteria: the accuracy of its classification results and its ability to select features. Our experiments demonstrate that the recognition performance of PFCVM$_{\text{LP}}$ on EEG emotion recognition datasets is either the best or close to the best. On high-dimensional gene expression datasets, PFCVM$_{\text{LP}}$ performs more accurately when compared to other approaches. A Rademacher complexity bound is derived for PFCVM$_{\text{LP}}$. By tightening this bound, we demonstrate the significance of feature selection and introduce a way of finding proper initial values.

PFCVM$_{\text{LP}}$ jointly encourages sparsity to features and samples. However, in order to select features for non-linear basis functions, we have to differentiate, which leads to high computational costs. As future work, we plan to use incremental learning [9, 15, 30] to reduce the computational costs. We also plan to design an online strategy [44] for joint feature and classifier learning. Also, PFCVM$_{\text{LP}}$ focuses on the supervised binary classification task. It would be interesting to extend PFCVM$_{\text{LP}}$ to other tasks, e.g., multi-class classification [24, 42], ordinal regression [30], or semi-supervised learning [22]. Finally, we aim to use PFCVM$_{\text{LP}}$ in other areas of research, such as in bioinformatics problems and clinical diagnoses [52].

APPENDIX

Hyperparameter Optimization

In order to compute a complete Bayesian classifier, feature and classifier co-learning includes computing this formula:

$$(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \arg\max_{(\boldsymbol{\alpha}, \boldsymbol{\beta})} p(\mathbf{t} \mid \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathrm{S}) p(\boldsymbol{\alpha}) p(\boldsymbol{\beta}), \tag{24}$$

where we assume $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are mutually independent. Equation (24) could be iteratively maximized between $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. The re-estimation rules of $\boldsymbol{\alpha}$ have been derived by [9]. In this appendix, we focus on deriving the re-estimating rules for $\boldsymbol{\beta}$, which means that we need to compute the following equation:

$$\boldsymbol{\beta} = \arg\max_{\boldsymbol{\beta}} p(\mathbf{t} \mid \boldsymbol{\alpha}^{\mathrm{old}}, \boldsymbol{\beta}, \mathrm{S}) p(\boldsymbol{\beta}). \tag{25}$$

The hyperprior $p(\boldsymbol{\beta})$ follows the Gamma distribution, $p(\boldsymbol{\beta}) = \Pi_{k=1}^{M} \mathrm{Gam}(\beta_k \mid c, d)$, where c and d are the parameters of the Gamma distribution.

As discussed in Section 3.2, we calculating a closed form of the marginal likelihood is non-trivial. Using Bayesian rules, the marginal likelihood is expanded as follows:

$$p(\mathbf{t} \mid \boldsymbol{\alpha}^{\mathrm{old}}, \boldsymbol{\beta}, \mathrm{S}) = \frac{p(\mathbf{t} \mid \mathbf{w}, \boldsymbol{\theta}, \mathrm{S}) p(\mathbf{w} \mid \boldsymbol{\alpha}^{\mathrm{old}}) p(\boldsymbol{\theta} \mid \boldsymbol{\beta})}{p(\mathbf{w}, \boldsymbol{\theta} \mid \mathbf{t}, \boldsymbol{\alpha}^{\mathrm{old}}, \boldsymbol{\beta})}. \tag{26}$$

Applying approximate Gaussian distributions for the sample and feature posteriors, in Section 3.1, we can obtain $p(\mathbf{w}, \boldsymbol{\theta} \mid \mathbf{t}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \approx \mathcal{N}(\mathbf{u}_{\boldsymbol{\theta}}, \Sigma_{\boldsymbol{\theta}}) * \mathcal{N}(\mathbf{u}_{\mathbf{w}}, \Sigma_{\mathbf{w}})$. As a result, we maximize the logarithm of Equation (25):

$$\begin{aligned} L &= \log[p(\mathbf{t} \mid \boldsymbol{\alpha}^{\mathrm{old}}, \boldsymbol{\beta}) p(\boldsymbol{\beta})] \\ &= \log p(\boldsymbol{\theta} \mid \boldsymbol{\beta}) - \log N(\mathbf{u}_{\boldsymbol{\theta}}, \Sigma_{\boldsymbol{\theta}}) + \log p(\boldsymbol{\beta}) + \mathrm{const} \\ &= \frac{1}{2}(\boldsymbol{\epsilon}^T \mathbf{B}^{-1} \boldsymbol{\epsilon} + \log|\mathbf{B}| - \log|\mathbf{H} + \mathbf{B}|) + \sum_{k=1}^{M} (c \log \beta_k - d\beta_k) + \mathrm{const}, \end{aligned}$$

where const is independent of $\boldsymbol{\beta}$, $\boldsymbol{\epsilon} = (\mathbf{D}^T(\mathbf{t} - \boldsymbol{\sigma}) + \mathbf{k}_{\boldsymbol{\theta}})$ is an M-dimensional vector and $\mathbf{H} = \mathbf{D}^T \mathbf{C} \mathbf{D} + \mathbf{O}_{\boldsymbol{\theta}} - \mathbf{E}$ is an $M \times M$ matrix. Practically, the latter two terms will disappear if we set $c = d = 0$.

To compute the optimal $\boldsymbol{\beta}$, we first differentiate Equation (27):

$$\frac{\partial L}{\partial \beta_k} = -\frac{1}{2}\left(\frac{\epsilon_k^2}{\beta_k^2} - \frac{1}{\beta_k} + \frac{1}{\beta_k + h_k} - 2\frac{c}{\beta_k} + 2d\right), \tag{27}$$

where h_k denotes the kth diagonal elements of $\mathbf{H}$. Note that $u_{\theta,k}^2 = \frac{\epsilon_k^2}{\beta_k^2}$ and $\Sigma_{\theta,kk} = \frac{1}{\beta_k + h_k}$, shown in Equations (10) and (11). So setting Equation (27) equal to 0, we obtain the update formula for $\boldsymbol{\beta}$:

$$\beta_k^{\mathrm{new}} = \frac{2c + 1}{u_{\theta,k}^2 + \Sigma_{\theta,kk} + 2d}, \tag{28}$$

which is the same formula as the EM-based solution established by [29], and guarantees a local optimum. However, if using the methodology of Bayesian Occam's razor reported by MacKay in [32], we derive more efficient update rules as follows:

$$\beta_k^{\mathrm{new}} = \frac{\gamma_k + 2c}{u_{\theta,k}^2 + 2d}, \tag{29}$$

where $\gamma_k \equiv 1 - \beta_k \Sigma_{\theta,kk}$.

ACKNOWLEDGMENTS

We are grateful to the associate editor and the anonymous reviewers for the constructive feedback and suggestions.

REFERENCES

[1] Uri Alon, Naama Barkai, Daniel A. Notterman, Kurt Gish, Suzanne Ybarra, Daniel Mack, and Arnold J. Levine. 1999. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences* 96, 12 (1999), 6745–6750.

[2] Richard Ernest Bellman. 1961. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton.

[3] James O. Berger. 1985. *Statistical Decision Theory and Bayesian Analysis* (2nd. ed.). Springer.

[4] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer, New York.

[5] Paul S. Bradley and Olvi L. Mangasarian. 1998. Feature selection via concave minimization and support vector machines. In *Proceedings of the 15th International Conference on Machine Learning*. Vol. 98, 82–90.

[6] Rich Caruana and Alexandru Niculescu-Mizil. 2004. Data mining in metric space: An empirical analysis of supervised learning performance criteria. In *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Dining*. ACM, 69–78.

[7] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2, 3 (2011), 27. Retrieved from http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/ datasets/binary.html.

[8] Huanhuan Chen, Peter Tino, and Xin Yao. 2009. Probabilistic classification vector machines. *IEEE Transactions on Neural Networks* 20, 6 (2009), 901–914.

[9] Huanhuan Chen, Peter Tino, and Xin Yao. 2014. Efficient probabilistic classification vector machine with incremental basis function selection. *IEEE Transactions on Neural Networks and Learning Systems* 25, 2 (2014), 356–369.

[10] Rizwan A. Choudrey. 2002. *Variational Methods for Bayesian Independent Component Analysis*. Ph.D. Dissertation. University of Oxford.

[11] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning* 20, 3 (1995), 273–297.

[12] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7, 1 (2006), 1–30.

[13] Ruonan Duan, Jiayi Zhu, and Baoliang Lu. 2013. Differential entropy feature for EEG-based emotion classification. In *Proceedings of the 6th International IEEE/EMBS Conference on Neural Engineering*. 81–84.

[14] Richard O. Duda, Peter E. Hart, and David G. Stork. 2012. *Pattern Classification*. John Wiley & Sons.

[15] Anita C. Faul and Michael E. Tipping. 2002. Analysis of sparse Bayesian learning. In *Proceedings of the 16th Conference on Neural Information Processing Systems*. 383–390.

[16] 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. Caligiuri, C. D. Bloomfield, and E. S. Lander. 1999. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* 286, 5439 (1999), 531–537.

[17] Michael Grant, Stephen Boyd, and Yinyu Ye. 2008. CVX: Matlab software for disciplined convex programming. Retrieved from http://cvxr.com/cvx/doc/install.html.

[18] Bin Gu, Xingming Sun, and Victor S. Sheng. 2017. Structural minimax probability machine. *IEEE Transactions on Neural Networks and Learning Systems* 28, 7 (2017), 1646–1656.

[19] Shan He, Huanhuan Chen, Zexuan Zhu, Douglas G. Ward, Helen J. Cooper, Mark R. Viant, John K. Heath, and Xin Yao. 2015. Robust twin boosting for feature selection from high-dimensional omics data with label noise. *Information Sciences* 291 (2015), 1–18.

[20] Xiaofei He, Deng Cai, and Partha Niyogi. 2005. Laplacian score for feature selection. In *Proceedings of the 18th Advances in Neural Information Processing Systems*. 507–514.

[21] Kaizhu Huang, Haiqin Yang, Irwin King, Michael R Lyu, and Laiwan Chan. 2004. The minimum error minimax probability machine. *Journal of Machine Learning Research* 5, (Oct. 2004), 1253–1286.

[22] Bingbing Jiang, Huanhuan Chen, Bo Yuan, and Xin Yao. 2017. Scalable graph-based semi-supervised learning through sparse Bayesian model. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2758–2771.

[23] Alexandros Kalousis, Julien Prados, and Melanie Hilario. 2007. Stability of feature selection algorithms: A study on high-dimensional spaces. *Knowledge and Information Systems* 12, 1 (2007), 95–116.

[24] Balaji Krishnapuram, Lawrence Carin, Mario AT Figueiredo, and Alexander J Hartemink. 2005. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 6 (2005), 957–968.

[25] Balaji Krishnapuram, Lawrence Carin, and Alexander J Hartemink. 2003. Joint classifier and feature optimization for cancer diagnosis using gene expression data. In *Proceedings of the 7th Annual International Conference on Research in Computational Molecular Biology*. ACM, 167–175.

[26] Balaji Krishnapuram, AJ Harternink, Lawrence Carin, and Mario AT Figueiredo. 2004. A Bayesian approach to joint feature selection and classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 9 (2004), 1105–1111.

[27] James T Kwok, Ivor Wai-Hung Tsang, and Jacek M Zurada. 2007. A class of single-class minimax probability machines for novelty detection. *IEEE Transactions on Neural Networks* 18, 3 (2007), 778–785.

[28] Gert RG Lanckriet, Laurent El Ghaoui, Chiranjib Bhattacharyya, and Michael I Jordan. 2002. A robust minimax approach to classification. *Journal of Machine Learning Research* 3, (Dec. 2002), 555–582.

[29] Chang Li and Huanhuan Chen. 2014. Sparse Bayesian approach for feature selection. In *Proceedings of the IEEE Symposium on Computational Intelligence in Big Data*. IEEE, 1–7.

[30] Chang Li and Maarten de Rijke. 2018. Incremental sparse Bayesian ordinal regression. *Neural Networks* 106 (2018), 294–302.

[31] Yi Li, Colin Campbell, and Michael Tipping. 2002. Bayesian automatic relevance determination algorithms for classifying gene expression data. *Bioinformatics* 18, 10 (2002), 1332–1339.

[32] David JC MacKay. 1992. Bayesian interpolation. *Neural Computation* 4, 3 (1992), 415–447.

[33] Ron Meir and Tong Zhang. 2003. Generalization error bounds for Bayesian mixture algorithms. *The Journal of Machine Learning Research* 4 (2003), 839–860.

[34] Yalda Mohsenzadeh, Hamid Sheikhzadeh, and Sobhan Nazari. 2016. Incremental relevance sample-feature machine: A fast marginal likelihood maximization approach for joint feature selection and classification. *Pattern Recognition* 60 (2016), 835–848.

[35] Yalda Mohsenzadeh, Hamid Sheikhzadeh, Ali M Reza, Najmehsadat Bathaee, and Mahdi M Kalayeh. 2013. The relevance sample-feature machine: A sparse Bayesian learning approach to joint feature-sample selection. *IEEE Transactions on Cybernetics* 43, 6 (2013), 2241–2254.

[36] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. 1998. UCI repository of machine learning databases. Retrieved from http://www.ics.uci.edu/ mlearn/MLRepository.html.

[37] Minh Hoai Nguyen and Fernando De la Torre. 2010. Optimal feature selection for support vector machines. *Pattern Recognition* 43, 3 (2010), 584–591.

[38] Feiping Nie, Heng Huang, Xiao Cai, and Chris H. Ding. 2010. Efficient and robust feature selection via joint L2, 1-norms minimization. In *Proceedings of the Advances in Neural Information Processing Systems*. 1813–1821.

[39] Feiping Nie, Shiming Xiang, Yangqing Jia, Changshui Zhang, and Shuicheng Yan. 2008. Trace ratio criterion for feature selection. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*. Vol. 2, 671–676.

[40] Sarah Nogueira and Gavin Brown. 2016. Measuring the stability of feature selection. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 442–457.

[41] Hanchuan Peng, Fulmi Long, and Chris Ding. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 8 (2005), 1226–1238.

[42] Ioannis Psorakis, Theodoros Damoulas, and Mark Girolami. 2010. Multiclass relevance vector machines: Sparsity and accuracy. *IEEE Transactions on Neural Networks* 21, 10 (2010), 1588–1598.

[43] Shirish Krishnaj Shevade and S. Sathiya Keerthi. 2003. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics* 19, 17 (2003), 2246–2253.

[44] Pannaga Shivaswamy and Thorsten Joachims. 2015. Coactive learning. *Journal of Artificial Intelligence Research* 53 (2015), 1–40. DOI: https://doi.org/10.1613/jair.4539

[45] Michael E. Tipping. 2001. Sparse Bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research* 1 (2001), 211–244.

[46] Vladimir Naumovich Vapnik. 1998. *Statistical Learning Theory*, Vol. 2. Wiley, New York.

[47] Anthony J. Viera and Joanne M. Garrett. 2005. Understanding interobserver agreement: The kappa statistic. *Family Medicine* 37, 5 (2005), 360–363.

[48] Haishuai Wang, Peng Zhang, Xingquan Zhu, Ivor Wai-Hung Tsang, Ling Chen, Chengqi Zhang, and Xindong Wu. 2017. Incremental subgraph feature selection for graph classification. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2017), 128–142.

[49] Jing Wang, Meng Wang, Peipei Li, Luoqi Liu, Zhongqiu Zhao, Xuegang Hu, and Xindong Wu. 2015. Online feature selection with group structure analysis. *IEEE Transactions on Knowledge and Data Engineering* 27, 11 (2015), 3029–3041.

[50] Mike West, Carrie Blanchette, Holly Dressman, Erich Huang, Seiichi Ishida, Rainer Spang, Harry Zuzan, John A. Olson, Jeffrey R. Marks, and Joseph R. Nevins. 2001. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences* 98, 20 (2001), 11462–11467.

[51] Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso A. Poggio, and Vladimir Vapnik. 2000. Feature selection for SVMs. In *Proceedings of the Advances in Neural Information Processing Systems*. 668–674.

[52] Darren J. Wilkinson. 2007. Bayesian methods in bioinformatics and computational systems biology. *Briefings in Bioinformatics* 8, 2 (2007), 109–116.
[53] Xindong Wu, Kui Yu, Wei Ding, Hao Wang, and Xingquan Zhu. 2013. Online feature selection with streaming features. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 5 (2013), 1178–1192.
[54] Xindong Wu, Kui Yu, Hao Wang, and Wei Ding. 2010. Online streaming feature selection. In *Proceedings of the 27th International Conference on Machine Learning*. 1159–1166.
[55] Yue Wu, Steven C. H. Hoi, Tao Mei, and Nenghai Yu. 2017. Large-scale online feature selection for ultra-high dimensional sparse data. *ACM Transactions on Knowledge Discovery from Data* 11, 4 (2017), 48.
[56] Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Efficient online learning for multitask feature selection. *ACM Transactions on Knowledge Discovery from Data* 7, 2 (2013), 6.
[57] Kui Yu, Wei Ding, Dan A. Simovici, Hao Wang, Jian Pei, and Xindong Wu. 2015. Classification with streaming features: An emerging-pattern mining approach. *ACM Transactions on Knowledge Discovery From Data* 9, 4 (2015), 30.
[58] Kui Yu, Xindong Wu, Wei Ding, and Jian Pei. 2016. Scalable and accurate online feature selection for big data. *ACM Transactions on Knowledge Discovery from Data* 11, 2 (2016), 16.
[59] Weilong Zheng and Baoliang Lu. 2015. Investigating critical frequency bands and channels for EEG-based emotion recognition with deep neural networks. *IEEE Transactions on Autonomous Mental Development* 7, 3 (2015), 162–175. Retrieved from http://bcmi.sjtu.edu.cn/ seed/download.html.
[60] Weilong Zheng, Jiyi Zhu, and Baoliang Lu. 2017. Identifying stable patterns over time for emotion recognition from EEG. *IEEE Transactions on Affective Computing*. DOI: 10.1109/TAFFC.2017.2712143

Received November 2017; revised September 2018; accepted January 2019