Efficient Probabilistic Classification Vector Machine with Incremental Basis Function Selection

Huanhuan Chen, *Member, IEEE,* Peter Tiňo, and Xin Yao, *Fellow, IEEE*

***Abstract*—Probabilistic classification vector machine (PCVM) [5] is a sparse learning approach aiming to address the stability problems of relevance vector machine (RVM) for classification problems. Since PCVM is based on the Expectation Maximization (EM) algorithm, it suffers from sensitivity to initialization, convergence to local minima, and the limitation of Bayesian estimation making only point estimates. Another disadvantage is that PCVM was not efficient for large data sets. To address these problems, this paper proposes an efficient probabilistic classification vector machine (EPCVM) by *sequentially* adding or deleting basis functions according to the marginal likelihood maximization for efficient training. Due to the truncated prior used in EPCVM, two approximation techniques, i.e. Laplace approximation and expectation propagation, have been used to implement EPCVM to obtain full Bayesian solutions. We have verified Laplace approximation and expectation propagation with a hybrid Monte Carlo approach. The generalization performance and computational effectiveness of EPCVM are extensively evaluated. Theoretical discussions using Rademacher complexity reveal the relationship between the sparsity and the generalization bound of EPCVM.**

***Index Terms*—Bayesian Classification, Efficient Probabilistic Classification Model, Incremental Learning, Laplace Approximation, Expectation Propagation, Support Vector Machine.**

I. Introduction and Background

Support vector machine (SVM) [37] and kernel methods are among the most popular learning methods in the machine learning community. Although SVM performs well for a broad range of practical applications, and is widely regarded as the state-of-the-art approach, it suffers from several disadvantages [35], including the *non-probabilistic*, hard binary decisions, and the number of support vectors grows linearly with the size of the training set, which increases the computational complexity when the problem becomes large.

Relevance Vector Machine (RVM) is a probabilistic learning method that tries to tackle these problems of SVM. RVM introduces a zero-mean Gaussian prior over every weight w_i and makes use of Bayesian Automatic Relevance Determination (ARD) framework [24] to obtain a sparse solution. In RVM, all basis functions are included in the model and those irrelevant basis functions will be deleted step by step based on evidence maximization. The necessary training and optimization of the marginal likelihood function is typically much slower. Later on, a highly accelerated RVM [36] has been proposed by optimizing the marginal likelihood function to enable an efficient sequential addition and deletion of candidate basis functions.

Huanhuan Chen is with the USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI), School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, 230027, China. Huanhuan Chen, Peter Tino and Xin Yao are with The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, Birmingham B15 2TT, United Kingdom, email: {H.Chen, P.Tino, X.Yao}@cs.bham.ac.uk.

However, Chen et al. [5] pointed out that RVM is not robust to kernel parameters due to the inappropriate formulation that adopts zero-mean Gaussian prior over weights for both positive and negative classes in classification problems, hence some training points that belong to positive class ($y_i = +1$) may have negative weights and vice versa. Chen et al. [5] demonstrated that RVM is unstable with respect to kernel parameters and might lead to sub-optimal solutions evidenced by empirical and theoretical results.

Probabilistic classification vector machine (PCVM) [5] introduced the non-negative, left-truncated Gaussian prior for positive training points ($y_i = +1$) and the non-positive, right-truncated Gaussian prior for negative training points ($y_i = -1$). A closed form Expectation Maximization (EM) was used to get a maximum a posteriori (MAP) estimation of parameters in PCVM. However, there are several limitations for the EM implementation. First, EM algorithm is sensitive to initializations and may converge to a local minimum, which will degrade the generalization ability, especially when the investigated problems become large where there are more local minima existing. Second, the EM algorithm can only obtain a MAP estimation of parameters. The MAP estimation is a limit of Bayes estimators under the 0-1 loss functions, but generally not a Bayes estimator per se. Third, EM based PCVM begins by including *all* basis functions and then pruning those irrelevant basis functions iteratively. Therefore, the algorithm is not appropriate for large data sets due to the computational/memory cost.

In order to address these problems, in this paper we improve PCVM in the following two directions:

- We construct an efficient probabilistic classification vector machine (EPCVM) and approximate the posterior by Laplace approximation (EPCVM$_{Lap}$) and expectation propagation [26] (EPCVM$_{EP}$). By using the two integral approximation techniques, the solution is fully Bayesian, which automatically tackles the first two disadvantages of the EM algorithm. The accuracy of EPCVM$_{Lap}$ and EPCVM$_{EP}$ has been verified by Markov Chain Monte Carlo (MCMC) algorithm.
- We have improved the PCVM algorithm using marginal likelihood maximization. By incrementally maximizing marginal likelihood, EPCVM can sequentially include basis functions into the models iteratively. This makes EPCVM$_{Lap}$ computationally more efficient.

The contributions of this paper can be summarized as follows:

- Unlike SVM, the proposed algorithms, are *probabilistic* models, producing the probabilistic outputs for new test points.
- Compared with the original PCVM, the two proposed methods, EPCVM$_{Lap}$ and EPCVM$_{EP}$, based on the integral approximation, are not only more stable with respect to initialization, but also yield better generalization performance (on the variety of data sets used).
- By incrementally maximizing marginal likelihood, the methods introduced in this paper reduce the computational complexity of PCVM.
- Due to the sparseness-inducing prior, the model sparsity helps to control model complexity and reduce the computational complexity in the *test* stage.

In sparse classification algorithms, the model is typically regularized by some prior belief about the weights that promote their sparsity. Besides Gaussian prior, Laplace prior that leads to an L1-penalty, analogous to the LASSO penalty for regression [34], is another popular choice. Joint classifier and feature optimization (JCFO) [21] was one of these algorithms using Laplace prior. JCFO was able to optimize the classifier and select the proper feature subsets simultaneously. To promote sparseness, sparse probit classification algorithm [15] adopted the hierarchical prior, i.e. the Jeffreys prior, over the Laplace prior, whose main advantage was able to control the degree of sparseness without prior parameters. However, both JCFO and sparse probit classification were based on expectation maximization algorithm, and they might suffer from the disadvantages, i.e., sensitivity to initialization and convergence to local minima. The above two algorithms are based on the specification of sparseness inducing prior to the weight vectors in parametric models. The sparse model has emerged in ensemble approaches as well. For example, Sun and Yao [33] proposed a sparse learning algorithm through gradient boosting for learning large kernel problems. However, this approach does not produce probabilistic outputs as it employed a greedy forward selection criterion by simply choosing the basis vector with the largest absolute value in the current residual.

Besides the parametric Bayesian models using prior to impose sparseness, there are a number of sparse non-parametric Bayesian models, e.g. sparse online Gaussian processes (GP) [10] and the accelerated version: informative vector machine (IVM) [22]. Sparse online Gaussian processes combines a Bayesian online algorithm with a sequential construction of a relevant subsample of the data that fully specifies the prediction of the GP model. IVM accelerated the spare GP model by approximating a Gaussian process using forward selection with criteria based on information-theoretic principles.

The rest of this paper is organized as follows. Section II proposes the two EPCVM implementations, followed by the comparisons of MCMC, Laplace approximation and expectation propagation in Section III. The experimental results and analysis are reported in Section IV. Section V provides the theoretical discussions on sparsity and generalization. Finally, Section VI concludes the paper and presents future work.

II. Efficient Probabilistic Classification Vector Machine

In this section, we will present the model specification for EPCVM in Section II-A, then the prior over weight vectors will be discussed in Section II-B. Section II-C presents Laplace approximation based EPCVM$_{Lap}$ algorithm, and Section II-D details the expectation propagation based EPCVM$_{EP}$ algorithm.

A. Model Formulation

Consider binary classification and a data set of input-target training pairs $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$, where $y_i \in \{-1, +1\}$. In order to transfer linear outputs to probabilistic outputs, a link function should be chosen to allow a smooth transition between two classes. The EM implementation of PCVM [5] used probit link function, i.e.

$$\psi(a) = \int_{-\infty}^{a} N(t|0,1)dt,$$

where $\psi(a)$ is the Gaussian cumulative distribution. In order to be consistent, the probit link function is employed in this paper as well.

Derivations of Laplace approximation become easier when sigmoid link function is used. In this paper, we employ a popular approximation [2] by making use of the similarity between the logistic sigmoid function and the probit link function [3] (pages 218-220), i.e.

$$\sigma(\lambda a) = \frac{1}{1+e^{-\lambda a}} \approx \psi(a),$$

where $\lambda = \sqrt{8/\pi}$. The scaling factor λ is chosen to ensure that the probit function and the logistic sigmoid function have the same slope at the origin.

After incorporating the link function, the EPCVM model becomes:

$$p(y=1|\mathbf{x},\mathbf{w}) = \psi(\mathbf{x};\mathbf{w}) = \psi\left(\sum_{i=0}^{N} y_i w_i \phi_i(\mathbf{x},\mathbf{x}_i)\right). \quad (1)$$

where $y_i \in \{-1,+1\}$ is the label, y_0 and $\phi_0(\mathbf{x},\mathbf{x}_0)$ are set to 1 for convenience. We use y_i in Equation (1) to make sure w_i is non-negative. In the following, we denote $y_i\phi_i(\mathbf{x})$ by $\phi_{y_i}(\mathbf{x})$, i.e. Equation (1) will be represented as follows:

$$p(y=1|\mathbf{x},\mathbf{w}) = \psi\left(\sum_{i=0}^{N} w_i \phi_{y_i}(\mathbf{x},\mathbf{x}_i)\right).$$

In Equation (1), we use $\phi(\cdot)$ instead of $K(\cdot)$ to indicate that basis functions in EPCVM do not need to satisfy Mercer's condition[1].

B. Truncated Prior over Weights

Based on the PCVM formulation [5], a truncated Gaussian prior [8] is introduced for each weight w_i and a zero-mean

[1]It must be a continuous symmetric kernel of a positive integral operator, which can be relaxed slightly to include conditionally positive kernels [31].

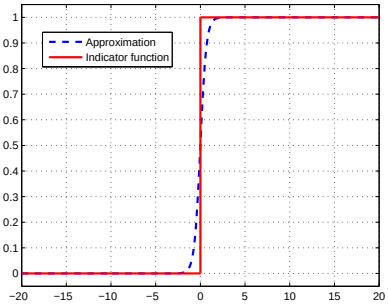


Fig. 1. Comparisons of indicator function and its differentiable approximation ξ_β ($\beta = 3$).

Gaussian prior is adopted for the bias w_0. The priors are assumed to be mutually independent.

$$\begin{aligned} p(\mathbf{w}|\alpha) &= \prod_{i=1}^{N} p(w_i|\alpha_i) = \prod_{i=1}^{N} N_t(w_i|0, \alpha_i^{-1}), \\ p(w_0|\alpha_0) &= N(w_0|0, \alpha_0^{-1}), \end{aligned}$$

where α_0 is the inverse variance, $N_t(w_i|0, \alpha_i^{-1})$ is a non-negative, left-truncated Gaussian, and α_i is the inverse variance. This is formalized in Equation (2):

$$\begin{aligned} p(w_i|\alpha_i) &= \begin{cases} 2N(w_i|0, \alpha_i^{-1}) & \text{if } w_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \\ &= 2N(w_i|0, \alpha_i^{-1}) \cdot \delta(w_i). \end{aligned} \tag{2}$$

where $\delta(\cdot)$ is the indicator function $\mathbf{1}_{x \geq 0}(x)$.

C. Laplace Approximation for EPCVM

In EPCVM, the prior is given as

$$p(\mathbf{w}|\alpha) = N(w_0|0, \alpha_0^{-1}) \prod_{i=1}^{N} 2N(w_i|0, \alpha_i^{-1}) \cdot \delta(w_i)$$

We follow convention and generalize the model by applying the logistic sigmoid link function, and adopting the Bernoulli distribution for $p(\mathbf{t}|\mathbf{w})$, the likelihood is written as follows:

$$p(\mathbf{t} \mid \mathbf{w}) = \prod_{n=1}^{N} \sigma_n^{t_n} \left[1 - \sigma_n\right]^{1 - t_n},$$

where $\sigma_n = \sigma\left(\lambda \sum_{i=0}^{N} w_i \phi_{y_i}(\mathbf{x}_n)\right)$ and we assume $y_0 = 1$ to facilitate the representation, $\mathbf{t} = [t_1, \cdots, t_N]^T$ is a vector of targets, $t_n = \frac{y_n + 1}{2} \in \{0, 1\}$ is the probabilistic target.

According to Bayes' theorem, the posterior distribution of weights $\mathbf{w}$ can be obtained with the current values of α as follows:

$$p(\mathbf{w}|\mathbf{t}) = \frac{p(\mathbf{t}|\mathbf{w}) p(\mathbf{w}|\alpha)}{p(\mathbf{t}|\alpha)}.$$

After incorporating the truncated Gaussian prior, the integral in Bayesian inference is intractable. In order to obtain the posterior, Laplace approximation will be employed to approximate the posterior. Laplace approximation is a deterministic approximation algorithm using a Gaussian to represent a given probability.

The most probable weight setting under the posterior, MAP estimate of $\mathbf{w}$, $\mathbf{w}_{MAP}$ can be obtained by maximizing the log of $p(\mathbf{w}|\mathbf{t})$ with respect to the parameters $\mathbf{w}$:

$$\begin{aligned} Q &= \log\left\{p(\mathbf{t}|\mathbf{w}) p(\mathbf{w}|\alpha)\right\} - \log p(\mathbf{t}|\alpha) \\ &= \sum_{n=1}^{N} \left[t_n \log \sigma_n + (1 - t_n) \log(1 - \sigma_n)\right] - \frac{1}{2} \sum_{i=0}^{N} \alpha_i w_i^2 \\ &\quad + \sum_{i=1}^{N} \log \delta(w_i) - \text{const.} \end{aligned}$$

As the indicator function $\delta(\cdot)$ is not differentiable, a sigmoid link function with $\beta = 3$ is employed to replace it, i.e. approximate $\delta(w_i)$ by $\xi_\beta(w_i) = \sigma(\beta w_i)$ (see Figure 1), the gradient is

$$\frac{\partial Q}{\partial \mathbf{w}} = \lambda \Phi^T (\mathbf{t} - \sigma) - \mathbf{A}\mathbf{w} + \mathbf{k},$$

where $\sigma = [\sigma_1, \cdots, \sigma_N]^T$, $\sigma_n = \sigma\left(\lambda \sum_{i=0}^{N} w_i \phi_{y_i}(\mathbf{x}_n)\right)$, $\mathbf{A} = diag(\alpha_0, \alpha_1, \cdots, \alpha_N)$ is the $(N+1) \times (N+1)$ diagonal matrix, $\mathbf{k} = [0, \beta(1 - \sigma(\beta w_1)), \cdots, \beta(1 - \sigma(\beta w_N))]^T$ is the $N + 1$ vector.

Setting the gradient to zero and we obtain

$$\mathbf{w}_{MAP} = \mathbf{A}^{-1} \left(\lambda \mathbf{\Phi}^T (\mathbf{t} - \sigma) + \mathbf{k}\right). \tag{3}$$

The Hessian can be explicitly computed as follows:

$$\frac{\partial^2 Q}{\partial \mathbf{w}^2} = -(\mathbf{\Phi}^T \mathbf{B} \mathbf{\Phi} + \mathbf{A} + \mathbf{D}),$$

where $\mathbf{B} = diag(b_1, \cdots, b_N)$ and $\mathbf{D}$ are diagonal matrices, where $b_i = \lambda^2 \sigma_n (1 - \sigma_n)$ and $\mathbf{D} = diag(0, d_1, \cdots, d_N) = diag(0, \sigma(\beta w_1)(1 - \sigma(\beta w_1))\beta^2, \cdots, \sigma(\beta w_N)(1 - \sigma(\beta w_N))\beta^2)$, respectively.

Hence, the posterior covariance is

$$\mathbf{\Sigma}_{MAP} = (\mathbf{\Phi}^T \mathbf{B} \mathbf{\Phi} + \mathbf{A} + \mathbf{D})^{-1}. \tag{4}$$

The novelty of the derivation in this section is to incorporate the indicator function, i.e. $\mathbf{k}$ and $\mathbf{D}$ in Equations (3) and (4), to prevent the weight from negative values, i.e. complying with truncated prior.

D. Expectation Propagation for EPCVM

Expectation propagation (EP) [26] is a deterministic *framework* to approximate Bayesian inference. It employs a family of exponential functions to minimize the KL-divergence between the exact term and the approximation term, and then EP combines these approximations analytically to obtain a Gaussian posterior. EP has been employed in various domains, such as Bayesian ensemble pruning [4], and multitask learning [30]. For a specific problem, such as EPCVM in this paper, plenty of derivations should be performed aiming to minimizing the KL-divergence between the exact term and the approximation term.

In EPCVM$_{EP}$, the likelihood for the weight vector $\mathbf{w}$ can be written as

$$p(y|\mathbf{x}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathbf{w}) = \prod_{n=1}^{N} \psi\left(y_n \sum_{i=0}^{N} w_i \phi_{y_i}(\mathbf{x}_n)\right).$$

By incorporating the prior with likelihood, the posterior of weight vectors $\mathbf{w}$ is calculated as

$$\begin{aligned} p(\mathbf{w}|\mathbf{x},y,\alpha) &\propto p(\mathbf{w}|\alpha)\prod_{n=1}^{N} p(y_n|\mathbf{x}_n,\mathbf{w}) \\ &= N(w_0|0,\alpha_0^{-1})\prod_{i=1}^{N} 2N(w_i|0,\alpha_i^{-1})\cdot \\ &\quad \prod_{i=1}^{N}\delta(w_i)\prod_{n=1}^{N} p(y_n|\mathbf{x}_n,\mathbf{w}). \end{aligned}$$

In EP, we need to approximate both the likelihood term $p(y_n|\mathbf{x}_n,\mathbf{w}) = \psi\left(y_n\sum_{i=0}^{N} w_i\phi_{y_i}(\mathbf{x}_n)\right)$ and $\delta(w_i)$ term. Denote the *exact* terms $g_n(\mathbf{w}) = p(y_n|\mathbf{x}_n,\mathbf{w})$ and $t_i(\mathbf{w}) = \delta(w_i) = \delta(\mathbf{w}^T\mathbf{e}_i)$ (where $\mathbf{e}_i = (0,\cdots,1,0,\cdots,0)^T$ is the standard basis to obtain the weight w_i ($w_i = \mathbf{w}^T\mathbf{e}_i$)) and the approximate terms by $\tilde{g}_n(\mathbf{w}) = s_n\exp\left(-\frac{1}{2v_n}(y_n\mathbf{w}^T\Phi(\mathbf{x}_n) - m_n)^2\right) = s_n\exp(-\frac{1}{2v_n}(\mathbf{w}^T\Phi_n - m_n)^2)$, where $\Phi(\mathbf{x}_n) = [\phi_{y_1}(\mathbf{x}_n),\cdots,\phi_{y_N}(\mathbf{x}_n)]^T$ and to simplify notation, $y_n\Phi(\mathbf{x}_n)$ is written as Φ_n, and $\tilde{t}_i(\mathbf{w}) = s_i\exp(-\frac{1}{2v_i}(\mathbf{w}^T e_i - m_i)^2)$. The EP for PCVM is described in the following.

1) Initialization the prior term: $q(\mathbf{w}) = p(\mathbf{w}|\alpha)$. Also initialize the approximating terms to 1: $\tilde{g}_n = 1$ and $\tilde{t}_i = 1$: $m = 0$, $v = \infty$ and $s = 1$.
2) Until both $\tilde{g}_n$ and $\tilde{t}_i$ converge: Loop $n = 1,\ldots,N$, and $i = 1,\ldots,N$;
   a) Remove the approximation term $\tilde{g}_n$ from the posterior $q(\mathbf{w})$ to obtain the leave-one-out posterior $q^{\setminus n}(\mathbf{w})$: $N(\mathbf{m}_{\mathbf{w}}^{\setminus n},\mathbf{V}_{\mathbf{w}}^{\setminus n})$.

$$\mathbf{V}_{\mathbf{w}}^{\setminus n} = \mathbf{V}_{\mathbf{w}} + \frac{(\mathbf{V}_{\mathbf{w}}\Phi_n)(\mathbf{V}_{\mathbf{w}}\Phi_n)^T}{v_n - \Phi_n^T\mathbf{V}_{\mathbf{w}}\Phi_n}, \quad (5)$$

$$\mathbf{m}_{\mathbf{w}}^{\setminus n} = \mathbf{m}_{\mathbf{w}} + (\mathbf{V}_{\mathbf{w}}^{\setminus n}\Phi_n)v_n^{-1}(\Phi_n^T\mathbf{m}_{\mathbf{w}} - m_n). \quad (6)$$

   b) Combine $q^{\setminus n}(\mathbf{w})$ and the exact term $g_n(\mathbf{w})$ to get $\hat{p}(\mathbf{w}) \propto q^{\setminus n}(\mathbf{w})g_n(\mathbf{w})$ and minimize the KL-divergence between $\hat{p}(\mathbf{w})$ and new posterior $q(\mathbf{w})$.

$$\begin{aligned} Z_n &= \int_{\mathbf{w}} q^{\setminus n}(\mathbf{w})g_n(\mathbf{w})d\mathbf{w} = \psi(z_n) \\ &= \psi\left(\frac{(\mathbf{m}_{\mathbf{w}}^{\setminus n})^T\Phi_n}{\sqrt{\Phi_n^T\mathbf{V}_{\mathbf{w}}^{\setminus n}\Phi_n + 1}}\right). \end{aligned}$$

   and

$$\begin{aligned} \mathbf{m}_{\mathbf{w}} &= \mathbf{m}_{\mathbf{w}}^{\setminus n} + \mathbf{V}_{\mathbf{w}}^{\setminus n}\frac{\partial\log Z_n}{\partial\mathbf{m}_{\mathbf{w}}^{\setminus n}} = \mathbf{m}_{\mathbf{w}}^{\setminus n} + \mathbf{V}_{\mathbf{w}}^{\setminus n}\Phi_n\rho_n \\ \mathbf{V}_{\mathbf{w}} &= \mathbf{V}_{\mathbf{w}}^{\setminus n} + \mathbf{V}_{\mathbf{w}}^{\setminus n}\begin{pmatrix} \frac{\partial\log Z_n}{\partial\mathbf{m}_{\mathbf{w}}^{\setminus n}}\left(\frac{\partial\log Z_n}{\partial\mathbf{m}_{\mathbf{w}}^{\setminus n}}\right)^T \\ -2\frac{\partial\log Z_n}{\partial\mathbf{V}_{\mathbf{w}}^{\setminus n}} \end{pmatrix}\mathbf{V}_{\mathbf{w}}^{\setminus n} \\ &= \mathbf{V}_{\mathbf{w}}^{\setminus n} + (\mathbf{V}_{\mathbf{w}}^{\setminus n}\Phi_n)\frac{\rho_n(\Phi_n^T\mathbf{m}_{\mathbf{w}} + \rho_n)}{\Phi_n^T\mathbf{V}_{\mathbf{w}}^{\setminus n}\Phi_n + 1}(\mathbf{V}_{\mathbf{w}}^{\setminus n}\Phi_n)^T, \end{aligned}$$

   where

$$\rho_n = \frac{1}{\sqrt{\Phi_n^T V_{\mathbf{w}}^{\setminus n}\Phi_n + 1}}\frac{N(z_n;0,1)}{\psi(z_n)}.$$

   c) Update the approximation term $\tilde{g}_n = Z_n\frac{q(\mathbf{w})}{q^{\setminus n}(\mathbf{w})}$, v_n, m_n and s_n are obtained as follows:

$$\begin{aligned} v_n &= \Phi_n^T\mathbf{V}_{\mathbf{w}}^{\setminus n}\Phi_n\left(\frac{1}{\rho_n(\Phi_n^T\mathbf{m}_{\mathbf{w}} + \rho_n)} - 1\right) \\ &\quad + \frac{1}{\rho_n(\Phi_n^T\mathbf{m}_{\mathbf{w}} + \rho_n)}, \\ m_n &= (\mathbf{m}_{\mathbf{w}}^{\setminus n})^T\Phi_n + (v_n + \Phi_n^T\mathbf{V}_{\mathbf{w}}^{\setminus n}\Phi_n)\rho_n, \\ s_n &= \psi(z_n)\sqrt{\Phi_n^T\mathbf{V}_{\mathbf{w}}^{\setminus n}\Phi_n v_n^{-1} + 1}\cdot \\ &\quad \exp\left(\frac{1}{2}\frac{\Phi_n^T\mathbf{V}_{\mathbf{w}}^{\setminus n}\Phi_n + 1}{\Phi_n^T\mathbf{m}_{\mathbf{w}} + \rho_n}\rho_n\right). \end{aligned}$$

   d) Remove the approximation term $\tilde{t}_i$ from the posterior $q(\mathbf{w})$ to obtain the leave-one-out posterior $q^{\setminus i}(\mathbf{w})$: $N(\mathbf{m}_{\mathbf{w}}^{\setminus i},\mathbf{V}_{\mathbf{w}}^{\setminus i})$. Refer to the equations (5) and (6).
   e) Combine $q^{\setminus i}(\mathbf{w})$ and the exact term $t_i(\mathbf{w})$ to get $\hat{p}(\mathbf{w}) = \frac{q^{\setminus i}(\mathbf{w})t_i(\mathbf{w})}{\int q^{\setminus i}(\mathbf{w})t_i(\mathbf{w})d\mathbf{w}}$ and minimize the KL-divergence $KL(\hat{p}(\mathbf{w})||q(\mathbf{w}))$ between $\hat{p}(\mathbf{w})$ and new posterior $q(\mathbf{w})$ subject to the constraint that $q(\mathbf{w})$ is a Gaussian distribution. Zeroing the gradient with respect to $\mathbf{m}_{\mathbf{w}}^{\setminus i}$ and $\mathbf{V}_{\mathbf{w}}^{\setminus i}$ gives the conditions [26],

$$\begin{aligned} E_{q(\mathbf{w})}[\mathbf{w}] &= E_{\hat{p}(\mathbf{w})}[\mathbf{w}], \\ E_{q(\mathbf{w})}[\mathbf{w}^T\mathbf{w}] &= E_{\hat{p}(\mathbf{w})}[\mathbf{w}^T\mathbf{w}]. \end{aligned}$$

This is the reason why the algorithm is named as expectation propagation.

$$\begin{aligned} Z_i &= \int_{\mathbf{w}} q^{\setminus i}(\mathbf{w})t_i(\mathbf{w})d\mathbf{w} \\ &= \int_{\mathbf{w}} q^{\setminus i}(\mathbf{w})\delta(\mathbf{w}^T\mathbf{e}_i)d\mathbf{w} = \psi(z_i) \end{aligned}$$

where

$$z_i = \frac{(\mathbf{m}_{\mathbf{w}}^{\setminus i})^T\mathbf{e}_i}{\sqrt{\mathbf{e}_i^T\mathbf{V}_{\mathbf{w}}^{\setminus i}\mathbf{e}_i}},$$

and

$$\begin{aligned} \frac{\partial\log Z_i}{\partial\mathbf{m}_{\mathbf{w}}^{\setminus i}} &= \frac{N(z_i)}{\psi(z_i)}\frac{\mathbf{e}_i}{\sqrt{\mathbf{e}_i^T\mathbf{V}_{\mathbf{w}}^{\setminus i}\mathbf{e}_i}} = \mathbf{g}_i \\ \frac{\partial\log Z_i}{\partial\mathbf{V}_{\mathbf{w}}^{\setminus i}} &= -\frac{1}{2}\rho_i\frac{(\mathbf{m}_{\mathbf{w}}^{\setminus i})^T\mathbf{e}_i}{\mathbf{e}_i^T\mathbf{V}_{\mathbf{w}}^{\setminus i}\mathbf{e}_i}\mathbf{e}_i\mathbf{e}_i^T = \mathbf{G}_i, \end{aligned}$$

where

$$\rho_i = \frac{N(z_i)}{\psi(z_i)}\frac{1}{\sqrt{\mathbf{e}_i^T\mathbf{V}_{\mathbf{w}}^{\setminus i}\mathbf{e}_i}}.$$

Based on the theory of expectation propagation [26],

$$\mathbf{m}_{\mathbf{w}} = \mathbf{m}_{\mathbf{w}}^{\setminus i} + \mathbf{V}_{\mathbf{w}}^{\setminus i}\frac{\partial\log Z_i}{\partial\mathbf{m}_{\mathbf{w}}^{\setminus i}} = \mathbf{m}_{\mathbf{w}}^{\setminus i} + \mathbf{V}_{\mathbf{w}}^{\setminus i}\rho_i\mathbf{e}_i,$$

$$\begin{aligned} \mathbf{V}_{\mathbf{w}} &= \mathbf{V}_{\mathbf{w}}^{\setminus i} + \mathbf{V}_{\mathbf{w}}^{\setminus i}(\mathbf{g}_i\mathbf{g}_i^T - 2\mathbf{G}_i)\mathbf{V}_{\mathbf{w}}^{\setminus i} \\ &= \mathbf{V}_{\mathbf{w}}^{\setminus i} + (\mathbf{V}_{\mathbf{w}}^{\setminus i}\mathbf{e}_i)\left(\frac{\rho_i\mathbf{e}_i^T\mathbf{m}_{\mathbf{w}}}{\mathbf{e}_i^T\mathbf{V}_{\mathbf{w}}^{\setminus i}\mathbf{e}_i}\right)(\mathbf{V}_{\mathbf{w}}^{\setminus i}\mathbf{e}_i)^T, \end{aligned}$$

f) Update the approximation term $\tilde{t}_i = Z_i \frac{q(\mathbf{w})}{q^{\setminus i}(\mathbf{w})}$:

$$v_i = \mathbf{e}_i^T \mathbf{V}_i \mathbf{e}_i = \mathbf{e}_i^T \mathbf{V}_{\mathbf{w}}^{\setminus i} \mathbf{e}_i \left(\frac{1}{\rho_i \mathbf{e}_i^T \mathbf{m}_{\mathbf{w}}} - 1 \right).$$

$$m_i = (\mathbf{m}_{\mathbf{w}}^{\setminus i})^T \mathbf{e}_i + (v_i + \mathbf{e}_i^T \mathbf{V}_{\mathbf{w}}^{\setminus i} \mathbf{e}_i)\rho_i,$$

$$s_i = \psi(z_i) \sqrt{\mathbf{e}_i^T \mathbf{V}_{\mathbf{w}}^{\setminus i} \mathbf{e}_i v_i^{-1} + 1} \exp\left(\frac{1}{2} \frac{\mathbf{e}_i^T \mathbf{V}_{\mathbf{w}}^{\setminus i} \mathbf{e}_i}{\mathbf{e}_i^T \mathbf{m}_{\mathbf{w}}} \rho_i \right).$$

Output: The approximated posterior of the weight vector $\mathbf{w}$

$$p(\mathbf{w}|\mathbf{x}, y, \alpha) \approx q(\mathbf{w}) = N(\mathbf{m}_{\mathbf{w}}, \mathbf{V}_{\mathbf{w}}).$$

Based on the algorithm, EP approximates each term as a Gaussian distribution, leading to the situation that the likelihood of every training point has similar forms as a regression likelihood term. The likelihood of each data point in EPCVM$_{EP}$ can be obtained as

$$p(\mathbf{m}|\mathbf{w}, \mathbf{x}) = (2\pi)^{-N} |\mathbf{\Lambda}|^{-1/2} \exp\left(\begin{array}{c} -\frac{1}{2}(\mathbf{w}^T \mathbf{\Phi} - \mathbf{m}_t)^T \Lambda^{-1} \cdot \\ (\mathbf{w}^T \mathbf{\Phi} - \mathbf{m}_t) \end{array} \right),$$

where $\mathbf{m}_t = (m_1, \cdots, m_N)$ denotes the target point vector, $\mathbf{\Lambda} = diag(v_1, \ldots, v_N)$, where v_n represents the variance of the noise for the training point n. EP actually maps a classification problem into a regression problem where (m_n,v_n) defines the virtual observation data point with mean m_n and variance v_n. Note that we can compute analytically the posterior distribution of the weights. The posterior distribution of the weight vector is thus given by:

$$\begin{aligned} p(\mathbf{w}|\mathbf{x}, \mathbf{m}, \alpha) &= \frac{p(\mathbf{m}|\mathbf{w}, \mathbf{x}) p(\mathbf{w}|\alpha)}{p(\mathbf{m}|\alpha, \mathbf{x})} \\ &= \frac{\exp\left(-\frac{1}{2}(\mathbf{w} - \mathbf{m}_{\mathbf{w}})^T \mathbf{V}_{\mathbf{w}} (\mathbf{w} - \mathbf{m}_{\mathbf{w}})\right)}{(2\pi)^N |\mathbf{V}_{\mathbf{w}}|^{1/2}}, \end{aligned}$$

where the posterior covariance and mean are:

$$\mathbf{V}_{\mathbf{w}} = (\mathbf{A} + \mathbf{\Phi} \mathbf{\Lambda}^{-1} \mathbf{\Phi}^T)^{-1}, \quad (7)$$

$$\mathbf{m}_{\mathbf{w}} = \mathbf{V}_{\mathbf{w}} \mathbf{\Phi} \mathbf{\Lambda}^{-1} \mathbf{m}_t. \quad (8)$$

where $\mathbf{A} = diag(\alpha_0, \cdots, \alpha_N)$.

1) Leave-one-out Estimation: A nice property of EP is that it can easily obtain an estimate of the leave-one-out error. In each iteration, EP computes the parameters of the approximate leave-one-out posterior $q^{\setminus n}(\mathbf{w})$ (step 2(a)) that does not depend on the n^{th} data point. So we can use the mean $\mathbf{m}_{\mathbf{w}}^{\setminus n}$ to approximate a classifier trained on the other $(N-1)$ data points. Thus an estimate of the leave-one-out error can be obtained as

$$err_{loo} = \frac{1}{N} \sum_{n=1}^{N} \delta(-y_n (\mathbf{m}_{\mathbf{w}}^{\setminus n})^T \Phi(\mathbf{x}_n)). \quad (9)$$

In practice, the estimation of leave-one-out error will be employed for model selection. The model with the smallest leave-one-out error will be selected instead of the one in the last iteration.

E. Hyperparameter Optimization for EPCVM

Originally, we optimized PCVM by the *top-down* approach [5]. It includes all basis functions in the beginning, and then prunes irrelevant basis functions when the corresponding $\alpha_n' s$ tending to infinity. However, the *top-down* approach will typically consume a lot of computational resources, especially in the beginning of the training. In order to make the algorithm more computationally efficient, we propose to use the *constructive* approach based on marginal likelihood maximization to include basis functions step by step starting from an empty model. This is different from the greedy forward selection criterion such as MAX-RES [33] that simply chooses the basis vector with the largest absolute value in the current residual.

The previous sections present the training algorithm of EPCVM$_{Lap}$ and EPCVM$_{EP}$ with fixed hyperparameter α. In order to sequentially update α for a practical algorithm, we can maximize the type-II marginal likelihood $p(D|\alpha)$. The fast algorithm to optimize the type-II marginal likelihood is to decompose $p(D|\alpha)$ into two parts, one part denoted by $p(D|\alpha_{\setminus i})$, that does not depend on α_i and another that does, i.e.,

$$p(D|\alpha) = p(D|\alpha_{\setminus i}) + l(\alpha_i), \quad (10)$$

where $l(\alpha_i)$ is a function that depends on α_i.

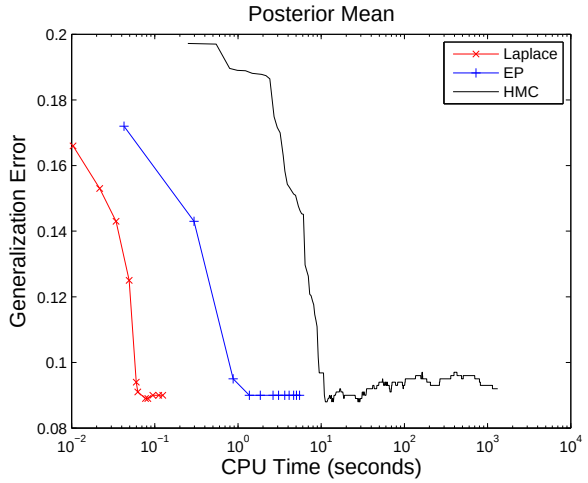
The updating rule for α_i can be obtained with the derivation of marginal likelihood [13]. The procedure leads to a practical algorithm for optimizing the hyperparameters that has significant speed advantages.

III. Laplace Approximation, Expectation Propagation and Markov Chain Monte Carlo

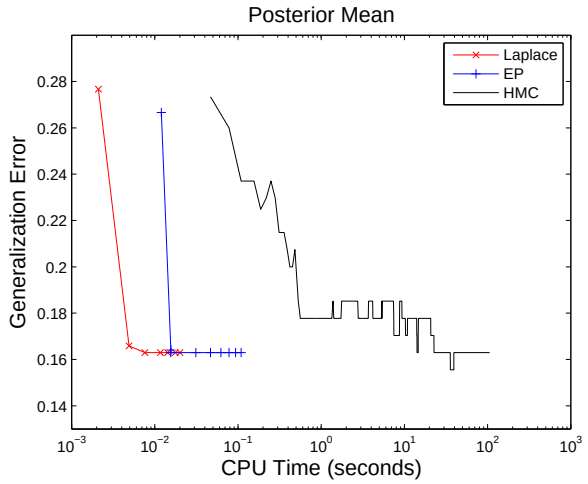
Laplace approximation and expectation propagation can be viewed as integral approximation techniques. As the truncated Gaussian prior is used in this paper, the exact posterior distribution is unknown. In this section, we employ Markov Chain Monte Carlo (MCMC) method to sample from the exact posterior distribution for the comparison with Laplace approximation and expectation propagation.

MCMC methods [1] are a class of algorithms for sampling from probability distributions based on constructing a Markov chain that has the desired distribution as its equilibrium distribution. MCMC may be too slow for many practical applications, but has the advantage that it becomes exact in the 'limit' of long runs. Thus, MCMC can provide a standard way to measure the accuracy of integral approximation methods, such as Laplace approximation and expectation propagation used in this paper. One popular MCMC algorithm, Metropolis algorithm [1] is sensitive to step size. The sampling result is slow and might exhibit a random-walk behavior with a small step size, whereas the result is inefficient due to a high rejection rate with a large step size.

This paper uses one powerful MCMC algorithm, hybrid Monte Carlo (HMC) algorithm [12] as it incorporates the gradient of the log probability with respect to the state variables into sampling process, which is able to make large changes to the system while keeping the rejection probability small.



(a) Synth



(b) Heart

Fig. 2. The comparisons of Laplace approximation, expectation propagation and hybrid monte carlo (200,0000 sampling points) in terms of generalization error and CPU time.

In our experiments, two data sets, synth[2] and heart [27], have been employed in the investigation. In Figure 2, we illustrate the comparisons of the three algorithms, i.e. Laplace, EP and HMC (200,000 sampling points), in terms of the generalization and the computational time. To compare the three algorithms, we do not optimize the hyperparameters in this figure[3]. The same random initialization is given to the three algorithms. According to Figure 2, Laplace, EP and HMC achieve similar performance. Due to the sampling mechanics, HMC converges slower than Laplace and EP, and in Synth data, the solution of HMC is unstable compared to Laplace and EP. The Laplace uses the least time and HMC has consumed the most time.

In the following experiments, we report the experimental results by incorporating the three algorithms with the hyperparameter optimization by maximizing the marginal likelihood.

[2]http://www.stats.ox.ac.uk/pub/PRNN/

[3]The following experiments will report the three algorithms with the hyperparameter optimization by maximizing the marginal likelihood, see Table I.

TABLE I
COMPARISONS OF MCMC, EP AND LAPLACE APPROXIMATION ON FOUR DATA SETS.

Methods	Cancer				Diabetics			
	error	AUC	#vec	CPUTime	error	AUC	#vec	CPUTime
MCMC	26.61	71.94	12	669.1s	23.17	82.86	23	764.1s
EP	26.65	72.53	9	3.2s	23.18	82.89	17	357.2s
Laplace	26.71	72.03	16	0.2s	23.11	83.12	22	1.1s
Methods	Heart				Thyroid			
	error	AUC	#vec	CPUTime	error	AUC	#vec	CPUTime
MCMC	16.37	90.67	16	707.4s	4.94	98.71	22	913.1s
EP	16.65	90.91	13	254.7s	5.16	98.63	10	61.2s
Laplace	16.65	90.83	15	0.3s	5.02	98.87	21	0.2s

In HMC, the posterior mean and covariance matrix are estimated using the sampling points. The same hyperparameter optimization procedures described in Section II-E are employed in HMC. The four data sets, including Cancer, Diabetics, Heart and Thyroid, from UCI machine learning repository are employed to show the difference between HMC, Laplace approximation and expectation propagation. The summary of these data sets can be referred in Table II. The resulting model and the classification performance are shown in Table I.

From the table, Laplace approximation, expectation propagation and MCMC achieve similar performances in terms of both generalization error and model size. Of course, Laplace approximation is much less time consuming than HMC.

Both figures and table indicate that Laplace approximation and EP approximate the posterior well with truncated Gaussian priors for the two classification problems. It also demonstrates that Laplace approximation is more efficient than EP and HMC in the current experimental settings. In the following section, we will conduct extensive experiments to compare Laplace approximation, EP and other algorithms.

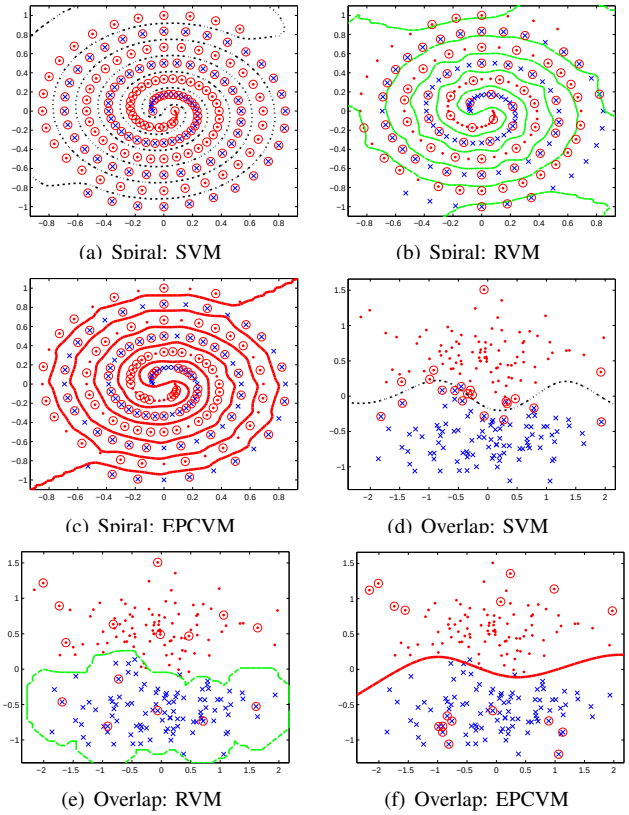
IV. EXPERIMENTAL STUDIES

First, we present experimental results of EPCVM, RVM and SVM on two synthetic data sets in order to understand the behaviors of these algorithms. Second, we carry out extensive experiments on 13 benchmark data sets using the error rate (ERR) and the area under the curve of receiver operating characteristic (AUC). Then, we present detailed statistical tests over multiple data sets for multiple classifiers. Finally, the algorithmic complexity of EPCVM and its application to a relatively large data set have been reported.

A. Synthetic Data Sets

In the first experiment, we compare EPCVM$_{Lap}$, SVM [37] and RVM [35] on two synthetic data sets. In order to facilitate further reference, each data set will be named according to its characteristics. *Spiral* can only be separated by highly nonlinear decision boundaries. *Overlap* comes from two Gaussian distributions with equal covariance, and is expected to be separated by a linear plane. This experiment employs a Gaussian RBF kernel as the basis function.

The parameters of SVM including the regularization parameter C and the kernel parameter θ are selected by grid search



(a) Spiral: SVM (b) Spiral: RVM (c) Spiral: EPCVM (d) Overlap: SVM (e) Overlap: RVM (f) Overlap: EPCVM

Fig. 3. Comparison of classification of synthetic data sets using a RBF kernel. Two classes are shown as dots and crosses. The separating lines are obtained by projecting test data over a grid. Kernel and regularization parameters for SVM, RVM and EPCVM are obtained by 10-fold cross validation

with 10-fold cross validation[4]. The kernel parameters θ of EPCVM$_{Lap}$ and RVM are selected by 10-fold cross validation.

In Figures 3 we present the decision boundaries of three algorithms. We observe a similar performance of EPCVM$_{Lap}$ and SVM in the case of *Spiral*. RVM cannot obtain the correct decision boundary due to the highly non-linear data set. The failure indicates that the prior of RVM produces *excessive* sparseness in the outer part of data, leading the boundary biasing towards outer circle and hence producing errors. EPCVM$_{Lap}$ produces "nearly linear" decision boundary in *Overlap* and RVM gives analogously curving decision boundary, whereas SVM gives a more curved boundary. We also notice that SVM uses the largest number of support vectors and RVM uses the smallest number of support vectors. EPCVM$_{Lap}$ seems to have reasonable vectors to achieve the tradeoff between model size and performance.

The results of EPCVM$_{Lap}$ are promising on the two synthetic data sets. EPCVM$_{Lap}$ not only handles the data sets with a predominating linear decision boundary, e.g. *Overlap*, but also be applied to the highly non-linear data sets, e.g. *Spiral*.

[4]The ranges of cross validation search for SVM are $C \in \{1, 3, \cdots, 100\}$ and $\theta \in \{0.1, 0.3, \cdots, 10\}$ (The data has been normalized to unit standard deviation.) in both synthetic data sets and benchmark data sets. The same search range $\theta \in \{0.1, 0.3, \cdots, 10\}$ has been used for EPCVM and RVM in both synthetic data sets and benchmark data sets.

TABLE II
SUMMARY OF 13 BENCHMARK DATA SETS.

Data	No. Train	No. Test	Positive %	Negative %	Dim
Abalone	2089	2088	50.18%	49.82%	8
Banana	2650	2650	44.83%	55.17%	2
Cancer	132	131	29.28%	70.72%	9
Diabetics	384	384	34.90%	65.10%	8
German	500	500	30.00%	70.00%	20
Heart	135	135	44.44%	55.56%	13
Image	1043	1043	56.95%	43.05%	18
Ringnorm	3700	3700	49.51%	50.49%	20
Splice	1496	1495	44.93%	55.07%	60
Thyroid	108	107	30.23%	69.77%	5
Titanic	1101	1100	58.33%	41.67%	3
Twonorm	3700	3700	50.04%	49.96%	20
Waveform	2500	2500	32.94%	67.06%	21

B. Benchmark Data Sets

In order to evaluate the performance of EPCVM$_{Lap}$ and EPCVM$_{EP}$, we compare different algorithms on 13 well known benchmark problems. These data sets include one synthetic set (banana) along with 12 other real-world data sets from UCI [27] and DELVE[5]. The characteristics of the data set are summarized in Table II. We follow Rätsch's methodology [29] and convert every problem into binary classes, and randomly partition every data set into 100 training and testing instances. In addition, every instance is input-normalized dimension-wise to have zero mean and unit standard deviation.

These compared algorithms are: EM based PCVM (PCVM$_{EM}$) [5], Laplace approximation based EPCVM (EPCVM$_{Lap}$) and expectation propagation based EPCVM (EPCVM$_{EP}$), SVM [37], relevance vector machine (RVM) [35] and sparse multinomial logistic regression (SMLR) [20]. The methodology to optimize the parameters of these models will be presented below.

In order to compare with some baseline methods, we also examine the performance of linear/quadratic discriminant analysis (LDA/QDA) and k Nearest Neighbor (kNN), where the number of nearest neighbors k is selected by the parameter selection methodology (where k is selected from $\{1, 2, \cdots, 20\}$). The error rate (ERR) and the area under the curve of receiver operating characteristic (AUC) are used for evaluation of these algorithms.

The procedure of parameter optimization follows Rätsch's methodology [29], which trains the algorithm with each candidate parameter on the first five training partitions of a given data set and selects the model parameters to be the median over those five estimates.

In the case of SVM, we train SVM with a parametrical grid with different combinations of the kernel parameter θ and the regularization parameter C, on the first five realizations of the training data and then select the median of the resulting parameters.

The same methodology is applied to PCVM$_{EM}$, EPCVM$_{Lap}$, EPCVM$_{EP}$, RVM, SMLR and kNN. The only difference among them is that they need to optimize different parameters. For PCVM$_{EM}$, EPCVM$_{Lap}$, EPCVM$_{EP}$, RVM and SMLR, we need to optimize the kernel width parameter

[5]http://www.cs.toronto.edu/~delve/data/datasets.html

TABLE III
COMPARISON OF kNN, LDA, QDA, SVM, RVM, SMLR AND PCVM$_{EM}$, EPCVM$_{Lap}$ AND EPCVM$_{EP}$ ON 13 BENCHMARK DATA SETS, BY % ERROR AND AUC. THESE RESULTS ARE THE AVERAGE OF 100 RUNS ON THE DATA SETS. "-" MEANS THE COVARIANCE MATRIX OF TRAINING DATA IS NOT POSITIVE DEFINITE AND QDA CANNOT OBTAIN THE RESULTS. BOLDFACE VALUES INDICATE THE BEST PERFORMANCE IN EACH DATA SET.

ERR	Abalone	Banana	Cancer	Diabetics	German	Heart	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
kNN	21.87	26.07	27.15	25.70	26.64	16.52	**4.17**	27.17	23.35	5.25	24.40	2.64	9.87
LDA	22.25	24.24	33.59	24.74	28.65	16.92	17.24	23.22	16.13	13.40	22.19	2.38	17.04
QDA	24.33	26.36	32.71	26.79	-	20.52	17.59	1.84	13.17	7.01	22.54	2.35	13.14
SVM	21.19	9.72	27.31	23.50	29.87	**16.15**	5.63	1.67	**10.41**	**4.57**	21.18	2.42	**8.86**
RVM	20.95	9.76	28.37	24.86	25.85	18.90	5.25	1.65	12.10	5.49	22.14	2.51	9.27
SMLR	20.98	9.74	27.39	23.39	25.06	19.52	5.89	1.68	11.41	5.25	21.18	2.32	9.21
EM	22.14	9.88	27.08	23.35	23.96	16.76	5.22	1.67	12.05	4.91	22.16	2.36	10.23
EP	21.16	**9.60**	**26.65**	23.18	**23.85**	16.65	5.16	1.66	11.27	5.16	21.02	**2.31**	9.16
Laplace	**20.42**	9.62	26.71	**23.11**	24.09	16.65	5.18	**1.52**	11.30	5.02	**20.83**	**2.31**	9.13
AUC	Abalone	Banana	Cancer	Diabetics	German	Heart	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
kNN	78.15	68.61	59.33	68.30	63.02	82.90	95.55	72.57	78.17	92.63	68.18	97.36	89.66
LDA	77.74	74.53	64.01	74.07	71.05	82.67	81.49	76.71	84.13	78.77	70.69	97.72	87.15
QDA	75.64	71.74	62.63	71.35	-	79.30	82.40	98.56	86.27	89.96	70.97	97.65	88.97
SVM	86.41	96.83	68.23	82.94	63.72	90.28	98.77	99.84	**95.94**	99.05	75.18	99.74	97.08
RVM	87.53	97.06	67.69	81.90	76.37	88.16	98.96	99.78	93.54	98.12	72.36	98.94	96.60
SMLR	**88.23**	96.89	70.89	81.85	77.58	87.16	99.03	99.75	95.37	98.36	74.80	99.71	**97.30**
EM	85.90	97.18	71.87	82.86	**79.95**	89.17	98.91	99.72	94.13	**99.17**	75.57	99.74	95.98
EP	87.52	**97.64**	**72.53**	82.89	79.87	**90.91**	**99.16**	99.86	95.88	98.63	77.87	**99.77**	97.14
Laplace	87.66	97.56	72.03	**83.12**	78.02	90.83	98.96	**99.91**	95.79	98.87	**77.89**	**99.77**	97.17

θ. For kNN, the number of nearest neighbors is selected by this methodology as well.

To select the best initialization point for PCVM$_{EM}$, we train PCVM$_{EM}$ with different initializations (8 initializations in this paper) over the first five training folds of each data set. Then we choose the best initialization point[6].

Table III reports the performance of these algorithms on the 13 benchmark data sets with ERR and AUC. According to this table, the performance of EPCVM$_{Lap}$ and EPCVM$_{EP}$ is similar. They outperform PCVM$_{EM}$ in terms of accuracy and AUC. EPCVM$_{Lap}$ wins 11 times over the metrics ERR and AUC, respectively, of them seven and four wins for ERR and AUC are significant, respectively.

In comparisons with other algorithms, EPCVM$_{Lap}$ and EPCVM$_{EP}$ perform very well in terms of two different metrics. For example, under the ERR metric it is observed that they outperforms all other methods in eight out of thirteen data sets, and comes second in three cases. They perform very well under the AUC metric, with the first place in eight cases and the second in the remaining four. Even when they fail under ERR metric on one of the data sets, e.g., Image, it can still win under the AUC metric. Although the RVM uses the Bayesian ARD framework, it seems that adopting the same prior for different classes leads to sub-optimal results.

The experimental results for SVM and SMLR are also enlightening. In most cases, the SVM and SMLR are worse than or comparable to the corresponding EPCVM$_{Lap}$ and EPCVM$_{EP}$. The baseline algorithms, kNN and LDA/QDA, only perform well on one data set. In all other cases, they fail to compete with the EPCVM and SVM, especially under the AUC metric.

Another interesting point is that EPCVM approaches achieve better performance by employing only a few of the data points, which has been illustrated by Table IV. In the three PCVM implementations, EPCVM$_{EP}$ tends to produce small models whereas EPCVM$_{Lap}$ often has larger models than EPCVM$_{EM}$ and EPCVM$_{EP}$. According to Table IV, the number of support vectors for SVM grows almost linearly with the number of training points, while RVM consistently uses much fewer data points. EPCVM$_{Lap}$ employs more vectors than the RVM but much less than SVM. This observation goes in accordance with the formulation. In RVM, the weights could reach zero from both sides because of the symmetrical zero-mean Gaussian, whereas the weights in EPCVM could only converge to zero from positive side because of the truncated Gaussian prior[7].

It is worth noting that the three PCVM algorithms have better performance than the RVM according to Table III. SMLR uses more vectors than three PCVM algorithms and RVM as it employed a cyclic component-wise update procedure [20], i.e., updating the weights even when they are deleted from the model, with a probability that is decreased with the number of iterations. This will be prone to include more basis functions into the model.

C. Statistical Comparisons on Single and Multiple Data Sets

In order to compare EPCVM$_{Lap}$ with other algorithms in a statistical context, we perform the statistical test for paired classifiers, e.g. EPCVM$_{Lap}$ vs. SVM and EPCVM$_{Lap}$ vs. RVM, on each single data set. We will carry out statistical tests on these two metrics and provide the win-loss-tie summary for these metrics. The threshold of the statistical t tests is set to be 0.05.

Table V gives the win-loss-tie summary of t-test based on 13 benchmark data sets. The significance tests show that under the

[6]With 8 initializations on first five training folds, we obtain an array of parameters of dimensions 8×5 where the rows are the initializations and the columns are the folds. For each column, we select the results that give the smallest test error, so that the array reduces from 40 to only 5 elements. Then we select the median over those parameters.

[7]The mean of truncated Gaussian prior is $\sqrt{\frac{2}{\pi\alpha_i}}$, which is not zero as normal Gaussian prior used in RVM.

TABLE IV
COMPARISON OF SVM, RVM, SMLR AND PCVM$_{EM}$, EPCVM$_{Lap}$ AND EPCVM$_{EP}$ ON 13 BENCHMARK DATA SETS, BY HOW MANY VECTORS AND STANDARD DEVIATION. THESE RESULTS ARE THE AVERAGE OF 100 RUNS ON THE DATA SETS.

Vectors	No. Train	SVM	RVM	SMLR	EM	EP	Laplace
Abalone	2089	1221.8±19.2	194.6±31.6	988.9±47.0	371.9±33.1	95.6±14.8	250.6±23.0
Banana	2650	606.6±19.5	264.6±37.6	1881.7±85.3	291.2±38.5	112.6±26.8	327.2±39.0
Cancer	132	117.4±12.1	11.2±2.5	72.8±6.4	12.4±2.6	8.3±2.1	15.9±4.0
Diabetics	384	380.1±2.6	16.3±5.0	205.3±17.4	17.6±5.1	16.3±3.1	21.5±4.8
German	500	478.4±4.3	32.9±8.6	193.8±13.1	26.8±7.1	31.7±8.1	63.3±9.2
Heart	135	127.3±3.6	12.1±2.4	25.7±5.0	6.2±2.1	11.7±4.1	14.8±2.3
Image	1043	728.8±11.5	24.3±3.2	511.8±13.9	158.3±21.7	181.2±16.9	200.3±12.7
Ringnorm	3700	3169±31.4	1728.8±27.7	1563.5±26.8	1421.8±23.8	1213.2±16.7	1849.3±14.4
Splice	1496	1496.0±0.0	167.2±86.3	504.3±25.1	153.2±29.3	118.9±16.3	275.9±15.9
Thyroid	108	54.7±3.6	30.8±21.9	43.7±4.6	8.3±2.1	9.7±2.3	20.4±4.7
Titanic	1101	489.5±17.2	61.4±21.1	826.5±57.3	137.8±31.4	116.9±9.9	121.1±11.8
Twonorm	3700	3216.0±6.9	769.2±29.3	2245.1±66.4	967.5±56.7	874.1±28.1	1018.6±32.3
Waveform	2500	2116.0±39.7	237.5±39.0	1180.5±46.3	837.5±41.2	281.6±32.5	538.2±42.7

TABLE V
STATISTICAL T TEST FOR 13 DATA SETS. FOR EACH METRIC, THE FIRST LINE IS THE WIN-LOSS-TIE SUMMARY OF THE ALGORITHM AGAINST THE EPCVM$_{Lap}$ BASED ON THE MEAN VALUE. THE SECOND ROW GIVES THE STATISTICAL SIGNIFICANCE WIN-LOSS-TIE SUMMARY BASED ON 13 BENCHMARK DATA SETS.

Data Sets	kNN	LDA	QDA	SVM	RVM	SMLR	EM	EP
ERR Mean	2-11-0	0-13-0	0-12-0	4-9-0	0-13-0	0-13-0	2-11-0	4-7-2
Significant	1-9-3	0-11-2	0-11-1	3-4-6	0-9-4	0-5-8	0-7-6	2-2-9
AUC Mean	0-13-0	0-13-0	0-12-0	2-11-0	0-12-1	3-10-0	2-11-0	5-7-1
Significant	0-13-0	0-13-0	0-12-0	0-5-8	0-10-3	1-5-7	1-4-8	2-0-11

TABLE VI
THE MEAN RANK OF THESE ALGORITHMS UNDER ERR AND AUC.

Rank	SVM	RVM	SMLR	EM	EP	Laplace
ERR	3.46	4.85	4.42	4.19	2.15	1.92
AUC	3.88	4.96	4.08	3.88	2.12	2.08

two metrics: a) PCVM$_{EM}$ never significantly win EPCVM$_{Lap}$ under ERR and it wins once and loses four times under AUC. EPCVM$_{EP}$ performs similar as EPCVM$_{Lap}$ under ERR: it wins twice and loses twice under ERR, and it slightly outperforms EPCVM$_{Lap}$ under AUC by winning twice and never loses. b) The differences between RVM and the EPCVM$_{Lap}$ are greater: RVM never wins under ERR and AUC. c) SVM wins three time and lose four times under ERR, and never wins under AUC. d) The experimental results also reveal that these baseline algorithms under-perform significantly against other algorithms.

In order to compare multiple algorithms based on multiple data sets, it is a common approach to count the number of times an algorithm performs better, worse or equal to the others. However, this method might not be reliable since it puts an arbitrary threshold of 0.05 or 0.10 on what counts and what does not for each data set [11]. Statistical tests on multiple data sets for multiple algorithms are preferred for comparing different algorithms over multiple data sets. In order to conduct statistical tests over multiple data sets, we perform the Friedman test [16] with the corresponding post-hoc tests. The Friedman test is a non-parametric equivalent of the repeated-measures analysis of variance (ANOVA) under the null hypothesis that all the algorithms are equivalent and so their ranks should be equal. This paper uses an improved Friedman test proposed by Iman and Davenport [17]. The

TABLE VII
FRIEDMAN TESTS WITH THE CORRESPONDING POST-HOC TESTS, BONFERRONI-DUNN, TO COMPARE CLASSIFIERS FOR MULTIPLE DATA SETS. THE THRESHOLD IS 0.10, AND $q_{0.10} = 2.326$.

Metrics	Friedman test	CD$_{0.10}$	SVM	RVM	SMLR	EM	EP
ERR	0.00	1.71	1.54	2.92	2.50	2.27	0.23
AUC	0.00	1.71	1.81	2.88	2.00	1.81	0.04

statistical test over multiple data sets has been used widely to evaluate the performance of classifiers e.g. [6], [7], [23], [32].

The Friedman test is carried out to test whether all the algorithms are equivalent. If the test result rejects the null hypothesis, i.e. these algorithms are equivalent, we can proceed to a post-hoc test. The power of the post-hoc test is much greater when all classifiers are compared with a control classifier and not among themselves. We do not need to make pairwise comparisons when we in fact only test whether a newly proposed method is better than the existing ones.

Based on this point, we would like to choose the EPCVM$_{Lap}$ as the control classifier to be compared with. Since the baseline classification algorithms are not comparable to SVM, RVM, SMLR, PCVM$_{EM}$, EPCVM$_{EP}$ and EPCVM$_{Lap}$, this section will analyze only six algorithms: SVM, RVM and SMLR, PCVM$_{EM}$ and EPCVM$_{EP}$ against the control classifier EPCVM$_{Lap}$.

The Bonferroni-Dunn test [11] is used as post-hoc tests when all classifiers are compared to the control classifier. The performance of pairwise classifiers is significantly different if the corresponding average ranks[8] differ by at least the critical difference (CD)

$$CD = q_\alpha \sqrt{\frac{j(j+1)}{6T}}, \tag{11}$$

where j is the number of algorithms, T is the number of data sets and critical values q_α can be found in [11]. For example, when $j = 6$, $q_{0.10} = 2.326$, where the subscript 0.10 is the threshold value.

[8]We rank these algorithms based on the metric on each data set and record the ranking of each algorithm as 1, 2 and so on. Average ranks are assigned in case of ties. The average rank of one algorithm is obtained by averaging over all of data sets. Please refer to Table VI for the mean rank of these algorithms under different metrics.

Table VI lists the mean rank of these algorithms under the two metrics: ERR and AUC. Table VII gives the Friedman test results. Since we employ the same threshold 0.10 for all three metrics, the critical difference $CD = 1.71$, where $j = 6$ and $T = 13$, is the same for these metrics. Several observations can be made from our results.

First, under the ERR metric, the differences between EPCVM$_{Lap}$ and RVM, SMLR, PCVM$_{EM}$, are greater than the critical difference, so the differences are significant, which means the EPCVM$_{Lap}$ is significantly better than RVM, SMLR and PCVM$_{EM}$ in this case. We could not detect any significant differences between SVM and EPCVM$_{EP}$. The correct statistical statement would be that *the experimental data are not sufficient to reach any conclusion regarding the difference between* EPCVM$_{Lap}$ *and SVM*/EPCVM$_{EP}$.

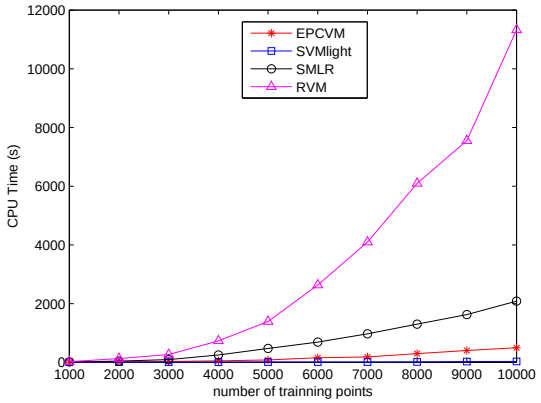
Second, EPCVM$_{Lap}$ significantly outperforms all other algorithms under the AUC metric except EPCVM$_{EP}$. Since the AUC metric requires relative accurate scores to discriminate positive and negative instances [14], EPCVM$_{Lap}$ succeeds by generating the probabilistic outputs. Another reason is that AUC is insensitive to the class skew/distribution [14] and some data sets used in this paper are imbalanced. In this way, EPCVM$_{Lap}$ and EPCVM$_{EP}$ perform well on these unbalanced data sets by considering different priors for different classes and thus have better scores under the AUC metric. SMLR generates point estimation based MAP, therefore it does not perform very well on AUC metric.

There are two major reasons why the two implementations of EPCVM, i.e. EPCVM$_{Lap}$ and EPCVM$_{EP}$, perform better than others.

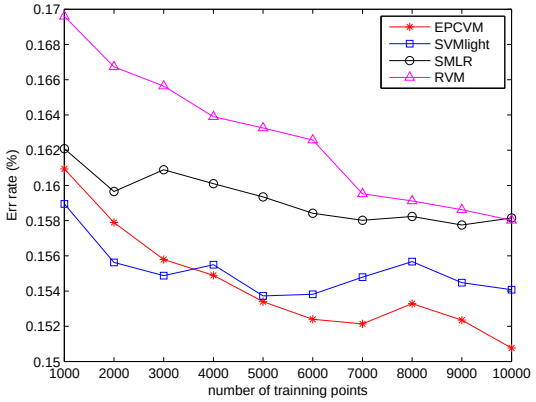
1) The robustness and sparseness are generated by the truncated Gaussian priors. These priors control the model complexity by including appropriate sparseness, and thus improve the model generalization.
2) As AUC prefers probabilistic outputs than hard decisions and it is insensitive to class unbalance, EPCVM$_{Lap}$ and EPCVM$_{EP}$ provide probabilistic outputs to assess the uncertainty for the predictions and perform well on these unbalanced data sets, which explain why EPCVM$_{Lap}$ and EPCVM$_{EP}$ are good under the AUC metric. Although the RVM also provides probabilistic outputs, it adopts Gaussian prior for training points belonging to both classes over weights and thus leads to inferior results.

D. Computational Complexity

The computational complexity and memory storage for PCVM$_{EM}$ is $O(N^3)$ and $O(N^2)$, respective, where N is the number of training points. The PCVM$_{EM}$ model will initially include *all*, i.e. N, basis functions in the beginning and reduce the model size gradually. This will leads to longer training times and larger memory usage. In addition, to address the common problems of EM, including sensitivity to initializations and convergence to local minima, the usual approach is to run the algorithm multiple times from different initialization points and choose the best one based on validation data, which will even increase the computational requirement in practice.



(a) CPU time on Adult Data Set



(b) Error Rate on Adult Data Set

Fig. 4. Comparison of CPU time and the error rate of EPCVM$_{Lap}$, SVM, SMLR and RVM on Adult data set.

In EPCVM$_{Lap}$, the update rules of $\mathbf{w}$ and b involve inversion of a matrix. The Cholesky decomposition is used in the practical implementation of the inversion to avoid numerical instability, which has the computational complexity $O(M^3)$ and memory storage $O(M^2)$, where M is the number of *non-zero* basis functions and $M << N$. In EPCVM$_{Lap}$, we will start when $M = 1$, and include basis functions step by step, i.e. increase M. As reported in Table IV, the final EPCVM$_{Lap}$ model usually has a small number of basis functions, i.e. a small M. This procedure will dramatically reduce the computational complexity.

Classical SVM algorithms has a time complexity of $O(N^3)$, where N is the number of training points, but the computational complexity of SVM can be reduced to approximately $O(N^{2.1})$ for sequential minimal optimization (SMO) like algorithms [28], which breaks the large quadratic programming (QP) problem into a series of smallest possible QP problems.

The training algorithm of SVM*light* has been optimized in many aspects [19], and it is even faster than the popular SMO algorithm for training SVM. The time complexity of each iteration in SVM*light* is $O(NDL)$, where D is the number of input features (input dimensionality) and L is a (regularization) parameter to control the number of rows of the Hessian to be computed in each iteration. The empirical

TABLE VIII
CPU TIME OF THE EPCVM$_{Lap}$ AND EPCVM$_{EP}$, PCVM$_{EM}$, SMLR, SVM, RVM, LDA, QDA, kNN ON 13 DATA SETS IN SECONDS. RESULTS ARE AVERAGED OVER 100 RUNS.

Time(s)	Abalone	Banana	Cancer	Diabetics	German	Heart	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
Laplace	29.36	22.11	0.23	1.06	3.04	0.28	18.59	147.76	43.03	0.21	1.92	49.37	33.57
EP	1004.90	1688.82	3.15	357.16	614.02	254.65	212.58	2127.43	384.75	61.18	33.10	1651.04	1481.51
EM	96.38	247.13	1.01	4.34	9.71	0.47	26.17	489.36	93.52	0.66	43.18	507.64	331.97
SVM	7.52	2.14	0.14	0.25	0.50	0.16	1.67	32.25	2.43	0.17	0.75	39.36	9.17
RVM	106.47	297.03	1.01	5.18	10.61	0.20	18.64	503.79	98.45	0.48	44.37	532.76	371.89
SMLR	68.83	51.76	0.27	1.79	4.84	0.19	10.96	156.14	25.51	0.50	3.00	196.55	108.01
LDA	0.08	0.01	0.06	0.00	0.00	0.06	0.00	0.02	0.16	0.00	0.00	0.03	0.00
QDA	0.09	0.01	0.00	0.06	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.05
kNN	0.13	0.21	0.00	0.00	0.06	0.00	0.09	0.86	0.27	0.00	0.06	0.56	0.73

time complexity of SVM*light* is $O(N^{1.7\sim2.0})$ [19], hence it is faster than EPCVM$_{Lap}$.

The computational complexity of EPCVM$_{EP}$ is $O(NM^3)$, which is the highest in the three implementations of PCVM. The long time consumed by EPCVM$_{EP}$ has been confirmed by Table VIII. In this paper, the aim to develop EPCVM$_{EP}$ is to confirm the effectiveness of EPCVM$_{Lap}$, and to compare the approximation accuracy of Laplace approximation. Since the computational complexity of EPCVM$_{EP}$ is higher, it is applicable for relatively small problems in the practical situations with the benefits to obtain compact models with fewer basis functions and the estimation of leave-one-out error in the training.

Table VIII shows the average CPU time of EPCVM$_{Lap}$, EPCVM$_{EP}$, PCVM$_{EM}$, SMLR, SVM, RVM, LDA, QDA, kNN on 13 data sets in seconds. Results are averaged over 100 runs. Note that in Table VIII, we do not record the cross validation time for parameter optimization in these algorithms.

To further study the computational effectiveness of EPCVM$_{Lap}$[9], SVM, SMLR and RVM, a relatively large data set, Adult from UCI machine learning repository, has been employed.

In Figure 4, the CPU time and the error rate of these algorithms on Adult data have been reported. As SVM*light* [19] has been used to implement SVM, in which sequential minimal optimization algorithm (SMO) and the optimization for large problems have been implemented. This is the reason why SVM*light* is the fastest algorithm. EPCVM$_{Lap}$ is programmed in Matlab and there is still room to improve its computational complexity by using C.

RVM and SMLR do not scale well with increased data points. SMLR employed a cyclic component-wise update procedure [20] and it will consider the weights even when they are deleted from the model. Therefore, the computation time becomes higher.

Figure 4 confirmed the computational effectiveness and the performance of EPCVM$_{Lap}$, as it scales well with the number of training points without compromising the performance.

The computational environment is Windows 7 with Intel Xeon QuadCore 3.10GHz CPU and 8GB RAM. The source codes of RVM and SMLR are obtained from Tipping's website[10], and Princeton's multi-voxel pattern analysis toolbox[11], respectively. EPCVM$_{Lap}$ and EPCVM$_{EP}$ are implemented in MATLAB.

[9]The computational complexity of EPCVM$_{EP}$ is much higher than EPCVM$_{Lap}$, SVM, SMLR and RVM. Therefore, we do not report the performance of EPCVM$_{EP}$ on the Adult data set.

V. SPARSITY AND GENERALIZATION

In this section we use Rademacher complexity [25] to investigate the relationship between the generalization bound for EPCVM and model sparsity.

Rademacher complexity quantifies "complexity" of function classes. Let F be a class of real-valued functions defined on X. The empirical Rademacher complexity of a functional class F on a data set $D = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$ is defined as

$$\hat{R}_N(F, D) = \frac{2}{N} E_\varsigma \left[\sup_{f \in F} \left| \sum_{i=1}^{N} \varsigma_i f(x_i) \right| \right]$$

where $\varsigma = (\varsigma_1, \varsigma_2, \cdots, \varsigma_N)$ is a vector random variable with elements independent binary random Rademacher variables such that $P(\varsigma_i = +1) = P(\varsigma_i = -1) = 1/2$ for all ς_i.

Assume that there is a distribution $P(\mathbf{x}, y)$ that generates the data items (i.e. $P(\mathbf{x}, y)$ represents the environment producing the data). The data set $D = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$ is generated i.i.d. form from $P(\mathbf{x}, y)$, i.e. D is generated from the product distribution $G(D)$, $G = P^N$. The Rademacher complexity of F is then

$$R_N(F) = E_{G(D)} \left[\hat{R}_N(F, D) \right]. \tag{12}$$

Rademacher complexity can be used to formulate generalization bound, as illustrated by the following theorem.

Theorem 1: [20], [25] Given a dataset $D = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$, for posterior distribution $q(\mathbf{w})$ over the parameters $\mathbf{w}$ (see section II-D), let

$$f(\mathbf{x}, q) = E_{q(\mathbf{w})}[sign(\mathbf{w}^T \phi(\mathbf{x}))].$$

For $s > 0$, let $R_{emp}^{(s)}$ be the empirical loss defined as

$$R_{emp}^{(s)}[f, D] = \frac{1}{N} \sum_{n=1}^{N} l_s(y_n f(\mathbf{x}_n, q)),$$

where the loss function[12] $l_s(a) = min(1, max(0, 1 - a/s))$ is $(1/s)$-Lipschitz. Consider arbitrary scalars $\rho > 0$, $r > 0$. Then,

[10]http://www.miketipping.com/

[11]http://code.google.com/p/princeton-mvpa-toolbox/

[12]For wrong classifications ($a < 0$), the loss is equal to 1. For correct classifications, s plays the role of the classification margin - even if the classification is correct ($a > 0$), if a is bellow s, a linearly scaled penalty $1 - a/s$ is still applied. The loss is zero only for $a \geq s$.

for $\vartheta \in (0,1)$, with probability at least $1-\vartheta$ over draws of training sets from G, the following bound for generalization error holds:

$$P(yf(\mathbf{x},q) < 0) \leq R^{(s)}_{emp}[f,D] + \frac{2}{s}\sqrt{\frac{2\tilde{\rho}(q)}{N}} + \sqrt{\frac{\ln \log_r \frac{r\tilde{\rho}(q)}{\rho} + \frac{1}{2}\ln\frac{1}{\vartheta}}{N}}, \quad (13)$$

and

$$\tilde{\rho}(q) = r \cdot \max(KL(q||p), \rho), \quad (14)$$

where $KL(q||p)$[13] is the Kullback-Leibler divergence from the posterior q to the prior p over parameters $\mathbf{w}$.

Note that the prior is integral part of our model, its hyperparameter α is modified during training. The Bayesian predictions of our model are based on the posterior over the weights that is in turn obtained from the optimized prior α. In this section we will denote the initial and optimized hyperparameter by $\boldsymbol{\alpha}_0 = (\alpha_{0,1}, \alpha_{0,2}, \cdots, \alpha_{0,N})$ and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_M)$, respectively. Based on the theorem, the generalization bound of EPCVM is related to the empirical loss and $KL(q||p)$. Given the same empirical loss, the generalization bound is tight provided $KL(q||p)$ is small. In the following, we will investigate the term $KL(q||p)$.

A. Kullback-Leibler Divergence from Posterior to Prior

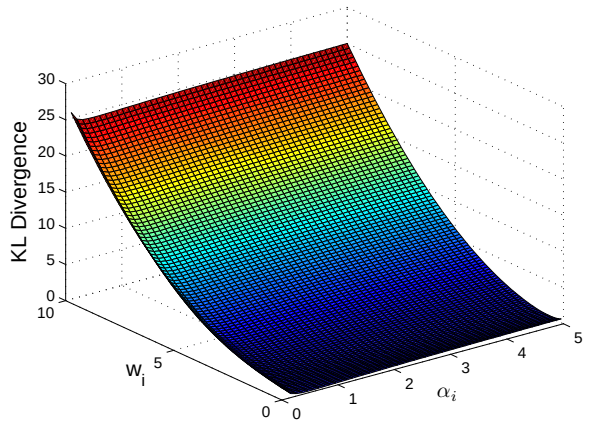


Fig. 5. An illustration of KL divergence between truncated posterior and truncated Gaussian prior.

The Kullback–Leibler (KL) divergence is a non-symmetric measure of the difference between two probability distributions. In this paper, posterior over weights is obtained through Laplace approximation $\tilde{q}(\mathbf{w})$. This posterior, $\tilde{q}(\mathbf{w})$, is a multivariate Gaussian with unbounded support. However, the prior is truncated to positive quadrant. The probability mass of $\tilde{q}$ in the positive quadrant is $A_0 = \int_0^\infty \tilde{q}(\mathbf{w})d\mathbf{w}$. Provided A_0 is sufficiently high, we can approximate $\tilde{q}$ by its renormalized version with support in the positive quadrant, $q(\mathbf{w}) = \tilde{q}(\mathbf{w})/A_0$.

[13]known as Bayesian surprise [18]

The KL divergence from $q(\mathbf{w})$ to prior $p(\mathbf{w}|\boldsymbol{\alpha}_0)$ can be calculated as

$$\begin{aligned} KL(q(\mathbf{w}) \parallel p(\mathbf{w}|\boldsymbol{\alpha}_0)) &= \int_0^\infty \frac{\tilde{q}(\mathbf{w})}{A_0}\ln\frac{\tilde{q}(\mathbf{w})}{A_0}d\mathbf{w} \\ &\quad - \int_0^\infty \frac{\tilde{q}(\mathbf{w})}{A_0}\ln p(\mathbf{w}|\boldsymbol{\alpha}_0)d\mathbf{w} \\ &= \frac{1}{A_0}\int_0^\infty \tilde{q}(\mathbf{w})\ln\frac{\tilde{q}(\mathbf{w})}{p(\mathbf{w}|\boldsymbol{\alpha}_0)}d\mathbf{w} \\ &\quad - \ln A_0. \end{aligned}$$

In this paper, we follow [20] and adopt the independence assumption on the posterior. Then (see Appendix A.5 of [9]),

$$\begin{aligned} D_{KL} &= KL(q(\mathbf{w}) \parallel p(\mathbf{w}|\boldsymbol{\alpha}_0)) \quad (15) \\ &= \sum_{i,w_i\neq 0} \left\{ \begin{array}{c} \frac{1}{2}\left[\frac{\alpha_{0,i}}{\alpha_i} - 1 + \ln\left(\frac{\alpha_i}{\alpha_{0,i}}\right) + \alpha_{0,i}w_i^2\right] \\ + \frac{(2\pi\alpha_i)^{-1/2}(\alpha_{0,i}+\alpha_i)w_i}{erfcx\left(-w_i\sqrt{\alpha_i/2}\right)} \\ - \ln\left(erfc\left(-\frac{w_i\alpha_i}{2}\right)\right) \end{array} \right\}, \end{aligned}$$

where $erfcx(a) = e^{a^2}erfc(a)$.

Since

$$A_{0,i} = \int_0^\infty \tilde{q}(w_i)dw_i = \frac{1}{2}erfc\left(-w_i\sqrt{\frac{\alpha_i}{2}}\right),$$

D_{KL} can be rewritten as follows:

$$D_{KL} = \sum_{i,w_i\neq 0} \left\{ \begin{array}{c} \frac{1}{2}\left[\frac{\alpha_{0,i}}{\alpha_i} - 1 + \ln\left(\frac{\alpha_i}{\alpha_{0,i}}\right) + \alpha_{0,i}w_i^2\right] \\ + \frac{(2\pi\alpha_i)^{-1/2}(\alpha_{0,i}+\alpha_i)w_i}{2\exp\left(\alpha_i w_i^2/2\right)}A_{0,i}^{-1} - \ln\left(A_{0,i}\right) \\ + \ln\left(\frac{erfc\left(-w_i\sqrt{\alpha_i/2}\right)}{2\cdot erfc(-w_i\alpha_i/2)}\right) \end{array} \right\}.$$

To show the characteristics of the KL divergence, in Figure 5 we illustrate the contributions of individual terms by fixing the initial hyperparameter priors to $\alpha_{0,i} = 0.5$ (the value used in our experiments). Two observations can be made: First, D_{KL} is much more sensitive to weight values w_i than to the optimized hyperparameters α_i. Second, D_{KL} is minimized for vanishing weights w_i. As discussed above, for comparable empirical errors, smaller D_{KL} is desirable. Therefore, employing truncated Gaussian priors in EPCVM to encourage sparsity by regularizing the weights to be smaller may have beneficial effects on the generalization, provided enough positive weights are preserved to ensure sufficient flexibility of the model.

Based on equations (13) and (15), the generalization bound of the EPCVM is a function of both the empirical loss term and the sparsity, represented by minimizing D_{KL}. According to Equation (14), trying to push D_{KL} to very small values beyond ρ is not desirable. Therefore, adequate sparsity is preferred in EPCVM, which matches our intuition regarding the nature of the generalization bound: If EPCVM chooses a non-sparse solution, the bounds might be loose; in contrast, if EPCVM chooses a proper sparse solution that can balance the empirical loss and the KL divergence D_{KL}, the bounds might be tight.

VI. Conclusion

In this paper, an efficient and effective probabilistic algorithm, EPCVM, was proposed for classification problems to improve a previous expectation maximization based PCVM algorithm [5]. The proposed algorithm addresses several previous limitations, including sensitivity to initializations, convergence to local minima, the solution being a point maximum-a-posterior (MAP) estimation, and unsuitability for large data sets.

The major improvements over PCVM$_{EM}$ are two folds. First, by employing Laplace approximation and expectation propagation, the solutions of EPCVM are fully Bayesian, which automatically tackle disadvantages of the EM algorithm, e.g., sensitivity to initializations, convergence to local minima, and the solution being a point MAP estimation. The accuracy of Laplace approximation and expectation propagation has been verified by Markov Chain Monte Carlo (MCMC) experiments, which give encouraging results. Second, by maximizing marginal likelihood, EPCVM$_{Lap}$ can sequentially include basis functions in the learned model step by step. This makes EPCVM$_{lap}$ computationally more efficient.

Our extensive empirical study confirms that EPCVM performs very well on the benchmark data sets under two metrics, especially under AUC. The difference between EPCVM and RVM shows that adopting truncated priors for different classes is beneficial. The difference between EPCVM$_{Lap}$ and PCVM shows that adopting Laplace approximation and sequential marginal likelihood maximization is beneficial in terms of generalization and computational efficiency. With higher computational complexity, EPCVM$_{EP}$ is applicable to relatively small problems with the benefits of more compact models with fewer basis functions and the estimation of leave-one-out error in the training.

From our results, we can conclude that the EPCVM$_{Lap}$ is a sparse learning algorithm that addresses drawbacks of SVM, RVM and PCVM$_{EM}$ without degrading the generalization performance. The number of basis functions in EPCVM$_{Lap}$ does not grow linearly with the number of training points, it is less and thus leads to simpler and easier-to-understand models. The theoretical analysis on generalization bounds using Rademacher complexity also supports the benefits of using truncated Gaussian prior to encourage sparsity in EPCVM. Future work for this study is to further increase the efficiencies of EPCVM and to extend the algorithm to multi-class classification problems.

Acknowledgment

This work is supported by the European Union Seventh Framework Programme under grant agreement No. INSFO-ICT-270428 on "Making Sense of Nonsense (iSense)". HC was supported by the National Natural Science Foundation of China under Grants 61203292, 61311130140 and the One Thousand Young Talents Program. PT was also supported from the Biotechnology and Biological Sciences Research Council grant [H012508/1]. XY was supported by a Royal Society Wolfson Research Merit Award.

References

[1] C. Andrieu, N. Freitas, A. Doucet, and M. Jordan, "An introduction to MCMC for machine learning," *Machine Learning*, vol. 50, no. 1–2, pp. 5–43, 2003.

[2] D. Barber and C. Bishop, "Ensemble learning for multi-layer networks," in *Advances in Neural Information Processing Systems*, vol. 10, 1998, pp. 395–401.

[3] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[4] H. Chen, P. Tino, and X. Yao, "Predictive ensemble pruning by expectation propagations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, pp. 999–1013, 2009.

[5] ——, "Probabilistic classification vector machines," *IEEE Transactions on Neural Networks*, vol. 20, pp. 901–914, 2009.

[6] H. Chen and X. Yao, "Regularized negative correlation learning for neural network ensembles," *IEEE Transactions on Neural Networks*, vol. 20, pp. 1962–1979, 2009.

[7] ——, "Multi-objective neural network ensembles based on regularized negative correlation learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1738–1751, 2010.

[8] H. Chen, P. Tino, and X. Yao, "A probabilistic ensemble pruning algorithm," in *Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, 2006, pp. 878–882.

[9] R. Choudrey, "Variational methods for bayesian independent component analysis," Ph.D. dissertation, University of Oxford, 2002.

[10] L. Csató and M. Opper, "Sparse on-line gaussian processes," *Neural Computation*, vol. 14, no. 3, pp. 641–668, 2002.

[11] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, pp. 1–30, 2006.

[12] S. Duane, A. Kennedy, B. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[13] A. Faul and M. Tipping, "Analysis of sparse bayesian learning," in *Advances in Neural Information Processing Systems 14*, 2002, pp. 383–389.

[14] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[15] M. Figueiredo, "Adaptive sparseness for supervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1050–1159, 2003.

[16] M. Friedman, "Comparison of alternative tests of significance for the problem of m rankings," *Annals of Mathematical Statistics*, vol. 11, pp. 86–92, 1940.

[17] R. Iman and J. Davenport, "Approximations of the critical region of the friedman statistic," *Communications in Statistics*, pp. 571–595, 1980.

[18] L. Itti and P. Baldi, "Bayesian surprise attracts human attention," in *Advances in Neural Information Processing Systems*, 2006, pp. 547–554.

[19] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds., 1999, pp. 169–184.

[20] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 957–968, 2005.

[21] B. Krishnapuram, A. Harternink, L. Carin, and M. Figueiredo, "A bayesian approach to joint feature selection and classifier design," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1105–1111, 2004.

[22] J. Langford, "Tutorial on practical prediction theory for classification," *Journal of Machine Learning Research*, vol. 6, no. 1, p. 273, 2006.

[23] N. Li, Y. Yu, and Z.-H. Zhou, "Diversity regularized ensemble pruning," in *Proceedings of the 23rd European Conference on Machine Learning (ECML'12)*, 2012, pp. 330–345.

[24] D. MacKay, "The evidence framework applied to classification networks," *Neural Computation*, vol. 4, no. 3, pp. 720–736, 1992.

[25] R. Meir and T. Zhang, "Generalization error bounds for bayesian mixture algorithms," *Journal of Machine Learning Research*, vol. 4, pp. 839–860, 2003.

[26] T. Minka, "Expectation propagation for approximate bayesian inference," in *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI'01)*, San Francisco, CA, USA, 2001, pp. 362–369.

[27] D. Newman, S. Hettich, C. Blake, and C. Merz, "UCI repository of machine learning databases," 1998, http://archive.ics.uci.edu/ml/.

[28] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.

[29] G. Rätsch, T. Onoda, and K. Müller, "Soft margins for adaboost," *Machine Learning*, vol. 42, no. 3, pp. 287–320, 2001.
[30] G. Skolidis and G. Sanguinetti, "Bayesian multitask classification with gaussian process priors," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2011–2021, 2011.
[31] A. J. Smola, B. Schölkopf, and K.-R. Müller, "The connection between regularization operators and support vector kernels," *Neural Networks*, vol. 11, pp. 637–649, 1998.
[32] R. G. F. Soares, H. Chen, and X. Yao, "Semi-supervised classification with cluster regularisation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, pp. 1779–1792, 2012.
[33] P. Sun and X. Yao, "Sparse approximation through boosting for learning large scale kernel machines," *IEEE Transactions on Neural Networks*, vol. 21, pp. 883–894, 2010.
[34] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society*, vol. 58, no. 1, pp. 267–288, 1996.
[35] M. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.
[36] M. Tipping and A. Faul, "Fast marginal likelihood maximisation for sparse bayesian models," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 2003, pp. 1–8.
[37] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, 1998.

Huanhuan Chen (M'09) received the B.Sc. degree from the University of Science and Technology of China, Hefei, China, in 2004, and Ph.D. degree, sponsored by Dorothy Hodgkin Postgraduate Award (DHPA), in computer science at the University of Birmingham, Birmingham, UK, in 2008. His PhD thesis "Diversity and Regularization in Neural Network Ensembles" has received 2011 IEEE Computational Intelligence Society Outstanding PhD Dissertation award (the only winner) and 2009 CPHC/British Computer Society Distinguished Dissertations Award (the runner up).

His work "Probabilistic Classification Vector Machines" on Bayesian machine learning published in IEEE Transactions on Neural Networks, has been awarded as IEEE Transactions On Neural Networks Outstanding Paper Award (bestowed in 2011, and only one paper in 2009 receive this award). His research interests include machine learning, data mining and evolutionary computation.

Peter Tiňo (M.Sc. Slovak University of Technology, Ph.D. Slovak Academy of Sciences) was a Fulbright Fellow with the NEC Research Institute, Princeton, NJ, USA, and a Post-Doctoral Fellow with the Austrian Research Institute for AI, Vienna, Austria, and with Aston University, Birmingham, U.K. Since 2003, he has been with the School of Computer Science, University of Birmingham, Edgbaston, Birmingham, U.K., where he is currently a Reader in complex and adaptive systems. His current research interests include dynamical systems, machine learning, probabilistic modeling of structured data, evolutionary computation, and fractal analysis. Peter was a recipient of the Fulbright Fellowship in 1994, the U.K.CHong-Kong Fellowship for Excellence in 2008, three Outstanding Paper of the Year Awards from the IEEE Transactions on Neural Networks in 1998 and 2011 and the IEEE Transactions on Evolutionary Computation in 2010, and the Best Paper Award at ICANN 2002. He serves on the editorial boards of several journals.

Xin Yao (F'03) is a Chair (Professor) of Computer Science and the Director of CERCIA (the Centre of Excellence for Research in Computational Intelligence and Applications), University of Birmingham, UK. He is an IEEE Fellow and a Distinguished Lecturer of IEEE Computational Intelligence Society (CIS). His work won the 2001 IEEE Donald G. Fink Prize Paper Award, 2010 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, 2010 BT Gordon Radley Award for Best Author of Innovation (Finalist), 2011 IEEE Transactions on Neural Networks Outstanding Paper Award, and many other best paper awards at conferences. He won the prestigious Royal Society Wolfson Research Merit Award in 2012 and was selected to receive the 2013 IEEE CIS Evolutionary Computation Pioneer Award. He was the Editor-in-Chief (2003-08) of IEEE Transactions on Evolutionary Computation. He has been invited to give more than 70 keynote/plenary speeches at international conferences. His major research interests include evolutionary computation and ensemble learning. He has more than 400 refereed publications in international journals and conferences.