Probabilistic Classification Vector Machines

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

***Abstract*—In this paper, a sparse learning algorithm, probabilistic classification vector machines (PCVMs), is proposed. We analyze relevance vector machines (RVMs) for classification problems and observe that adopting the same prior for different classes may lead to unstable solutions. In order to tackle this problem, a signed and truncated Gaussian prior is adopted over every weight in PCVMs, where the sign of prior is determined by the class label, i.e., $+1$ or -1. The truncated Gaussian prior not only restricts the sign of weights but also leads to a sparse estimation of weight vectors, and thus controls the complexity of the model. In PCVMs, the kernel parameters can be optimized simultaneously within the training algorithm. The performance of PCVMs is extensively evaluated on four synthetic data sets and 13 benchmark data sets using three performance metrics, error rate (ERR), area under the curve of receiver operating characteristic (AUC), and root mean squared error (RMSE). We compare PCVMs with soft-margin support vector machines ($\mathrm{SVM_{Soft}}$), hard-margin support vector machines ($\mathrm{SVM_{Hard}}$), SVM with the kernel parameters optimized by PCVMs ($\mathrm{SVM_{PCVM}}$), relevance vector machines (RVMs), and some other baseline classifiers. Through five replications of twofold cross-validation F test, i.e., 5×2 cross-validation F test, over single data sets and Friedman test with the corresponding post-hoc test to compare these algorithms over multiple data sets, we notice that PCVMs outperform other algorithms, including $\mathrm{SVM_{Soft}}$, $\mathrm{SVM_{Hard}}$, RVM, and $\mathrm{SVM_{PCVM}}$, on most of the data sets under the three metrics, especially under AUC. Our results also reveal that the performance of $\mathrm{SVM_{PCVM}}$ is slightly better than $\mathrm{SVM_{Soft}}$, implying that the parameter optimization algorithm in PCVMs is better than cross validation in terms of performance and computational complexity. In this paper, we also discuss the superiority of PCVMs' formulation using maximum *a posteriori* (MAP) analysis and margin analysis, which explain the empirical success of PCVMs.**

***Index Terms*—Bayesian classification, machine learning, probabilistic classification model, support vector machine.**

I. Introduction

In binary classification, we are given a set of input vectors $\{\mathbf{x}_i\}_{i=1}^N$ together with the corresponding class labels $\{y_i\}_{i=1}^N$, where $y_i \in \{1, -1\}$. The goal is to infer a function $f(\mathbf{x}; \mathbf{w})$ based on this training set. This can be done by choosing a learning model $f(\cdot)$ which is controlled by some unknown parameters $\mathbf{w}$, and "learning" these parameters from the given training set. The obtained classifier is evaluated by its generalization ability, i.e., how accurately it performs on new data assumed to follow the same distribution as the training data.

Manuscript received August 21, 2008; revised November 18, 2008; accepted January 19, 2009. First published April 24, 2009; current version published June 03, 2009.

The authors are with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: H.Chen@cs.bham.ac.uk; P.Tino@cs.bham.ac.uk; X.Yao@cs.bham.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2014161

Recently, the model, in which the prediction $f(\mathbf{x}; \mathbf{w})$ is expressed as a linear combination of basis functions $\phi_{\boldsymbol{\theta}}(\mathbf{x})$, has attracted much research interest [3], [26]

$$f(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{N} w_i \phi_{i,\boldsymbol{\theta}}(\mathbf{x}) + b = \boldsymbol{\Phi}_{\boldsymbol{\theta}}(\mathbf{x})\mathbf{w} + b \qquad (1)$$

where the weight vector $\mathbf{w} = (w_1, \cdots, w_N)^T$ is parameter of the model, b is the bias, and $\boldsymbol{\Phi}_{\boldsymbol{\theta}}(\mathbf{x}) = (\phi_{1,\boldsymbol{\theta}}(\mathbf{x}), \cdots, \phi_{N,\boldsymbol{\theta}}(\mathbf{x}))$ is the basis function vector, wherein θ is the parameter vector of the basis function. The learning algorithm is to adjust the parameters $\mathbf{w} = (w_1, \cdots, w_N)^T$, b, and θ to achieve a good generalization ability.

Among the range of model (1), support vector machines (SVMs) [27] are one of the most popular methods. SVMs make predictions based on the function

$$f(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{N} w_i K_\theta(\mathbf{x}, \mathbf{x}_i) + b = \mathbf{K}_\theta(\mathbf{x})\mathbf{w} + b \qquad (2)$$

where $\mathbf{K}_\theta(\mathbf{x}) = (K_\theta(\mathbf{x}, \mathbf{x}_1), \cdots, K_\theta(\mathbf{x}, \mathbf{x}_N))$ is the kernel function and the weight vector $\mathbf{w}$ is parameter of the model. Note that the SVM predictor is not defined explicitly in this form, rather (2) emerges implicitly as a consequence of the use of the kernel function to define a dot-product in some notional feature space.

The success of SVMs is attributed to the margin maximization theory [27]. The formulation of SVMs maximizes the margin between different classes, leading to a sparse model depending on the training points that either lie on the margin or on the wrong side of it.

Although an SVM performs well for a broad range of practical applications, and is widely regarded as the state-of-the-art approach, it suffers from the following disadvantages.

- *Nonprobabilistic* but hard binary decisions do not provide the uncertainty for predictions. The probabilistic predictions are particularly crucial in classification problems when posterior probabilities of class membership are adapted to varying class priors and asymmetric misclassification costs. The probabilistic predictions are also important for decision making. Some postprocessing techniques have been employed to transform the binary outputs to probabilistic outputs for SVMs. For example, Platt *et al.* [21] trained the parameters of an additional sigmoid function to map the SVMs outputs into probabilities. However, Tipping argued that these estimates are unreliable [26].
- The number of support vectors grows linearly with the size of the training set, which increases the computational complexity when the problem becomes large. Some postprocessing techniques are often required to reduce the computational complexity [4] of SVMs.

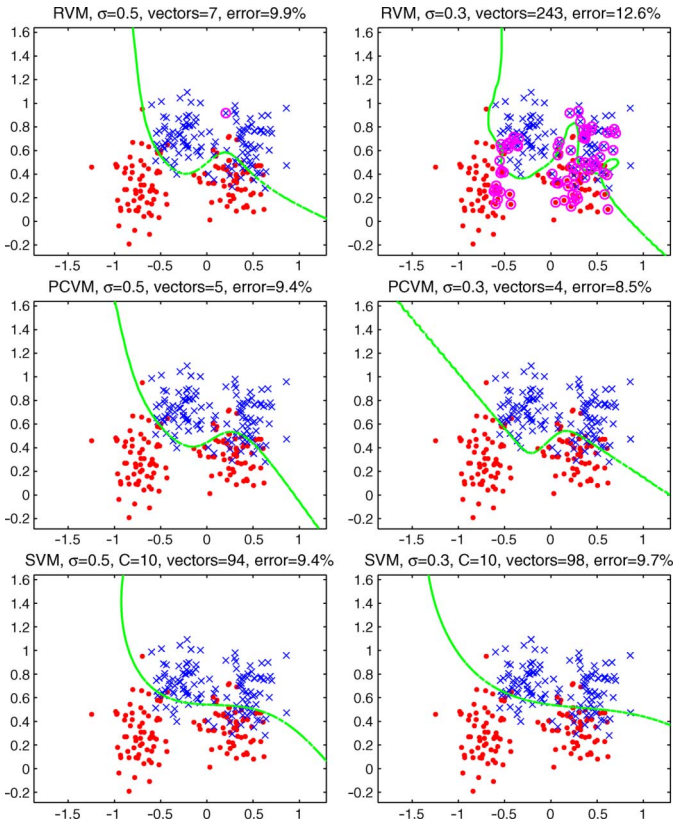


Fig. 1. Illustration of decision boundaries of RVM, PCVM, and SVM with same kernel parameters for Synth data set. The vectors whose weights have opposite signs are shown circled.

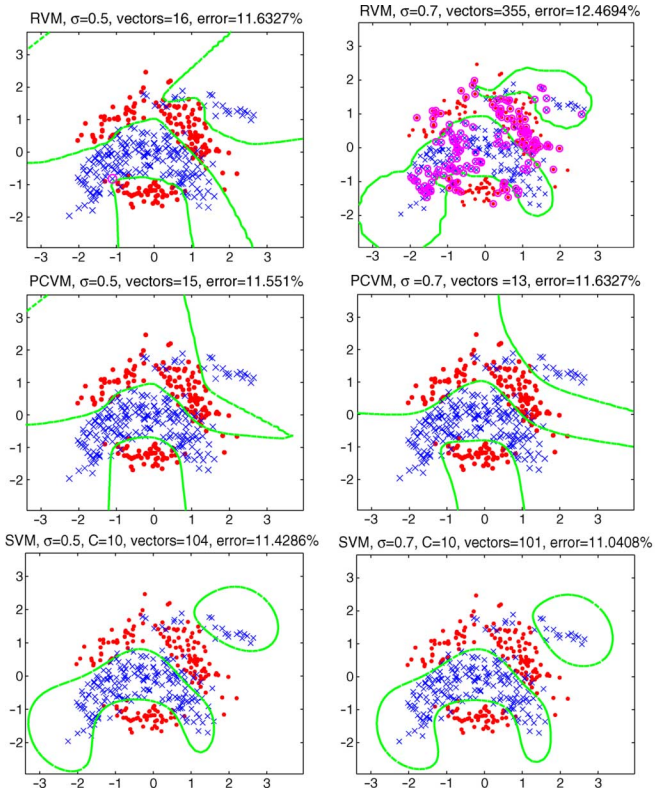


Fig. 2. Illustration of decision boundaries of RVM, PCVM, and SVM with same kernel parameters for Banana data set. The vectors whose weights have opposite signs are shown circled.

- Several parameters need to be tuned by cross validation. The parameters, including the error/margin tradeoff parameter C (a large C corresponding to assigning a higher penalty to errors) and the parameters of kernel function, are crucial for the performance of SVMs. Optimization of these parameters usually involves grid search by cross validation, whose computation is extremely expensive. Once the inappropriate range of search grid is adopted, the obtained parameters do not work and we have to respecify the search range and repeat the process.

In order to address these problems of SVMs, relevance vector machines (RVMs) have been proposed [26] to produce probabilistic predictions based on Bayesian techniques. RVMs introduce a zero-mean Gaussian prior over every weight w_i and make use of Bayesian automatic relevance determination (ARD) framework [17], [18] to obtain a sparse solution. As a result of sparseness-inducing prior, posteriors of many weights are sharply distributed around zero, hence these weights are pruned and the model becomes sparse.

However, RVMs [26] adopt the 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. This formulation might result in the situation that the decision of RVMs is based on some untrustful vectors, and thus is sensitive to the kernel parameter. Figs. 1 and 2 illustrate this phenomenon in RVMs.

The source code of RVM is directly downloaded from Tipping's website.[1] We utilize Ripley's Synth data set[2] and Rätsch's Banana data set[3] in Figs. 1 and 2. The Synth data were generated from mixtures of two Gaussians by Ripley [24], with the classes overlapping to the extent that the Bayesian error is around 8%. Banana is generated by Rätsch [23] with more complicated decision boundaries. In Rätsch's implementation, there are 100 folds in the Banana data set and Fig. 2 is based on the first fold. In both figures, the Gaussian radial basis function (RBF) has been used for SVMs and RVMs.

According to these figures, RVMs often utilize the vectors with opposite signs even with well-selected kernel parameters. Assume that "×" stands for positive class and "•" stands for negative class. In the first subfigure of Fig. 1, RVMs assign a negative weight to a positive vector that is in the heart of a positive area. Intuitively, it is unstable to trust this negative weight on the positive vector. When the kernel parameter is changed a little, in Fig. 1 from 0.5 to 0.3, RVMs utilize much more redundant vectors (243 out of 250, where almost half are with opposite weights) than SVMs and thus overfit the noise. The results are similar in Fig. 2.

Compared with RVMs, PCVMs and SVMs are more robust with respect to kernel parameters. PCVMs and SVMs always assign positive/negative vectors with positive/negative weights.

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

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

[3]http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm

This principle is implemented in SVMs by enforcing the Lagrange multipliers to be nonnegative. In (2), the weight vector is defined as $\mathbf{w} = (\upsilon_1 y_1, \cdots, \upsilon_N y_N)^T$, where υ_i's are nonnegative Lagrange multipliers and $y_i \in \{1, -1\}$ are the class labels. It means that $w_i = \upsilon_i y_i$ must have the same sign (some are zero) as the corresponding y_i.

However, as a probabilistic classification model, RVMs do not follow this principle and adopt a zero mean Gaussian for both classes, which facilitates the integral computation but results in suboptimal results.

In order to address this problem of RVMs and propose an appropriate probabilistic model for classification problems, this paper proposes a probabilistic algorithm, probabilistic classification vector machines (PCVMs), which introduces different priors over weights for training points belonging to different classes, i.e., the nonnegative, left-truncated Gaussian for the positive class $(y_i = +1)$ and the nonpositive, right-truncated Gaussian for the negative class $(y_i = -1)$. PCVMs also implement a parameter optimization procedure for kernel parameters in the training algorithm, which is proven to be effective in practice. As the integral is intractable in probabilistic inference with the truncated Gaussian prior, a closed-form expectation–maximization (EM) is used to get a maximum *a posteriori* (MAP) estimation of parameters.

Our approach not only addresses the issues concerned with SVMs, but also provides the following advantages. (1) Being a *probabilistic* model, the approach produces the probabilistic outputs for new test points. (2) The procedure for optimizing kernel parameters in the EM algorithm is effective and avoids the computationally expensive grid search by cross validation. (3) Because of the sparseness-inducing prior, the model generates adequate sparseness in the estimation of weight vector. The sparseness controls the complexity and reduces the computational complexity in the *test* stage.

The rest of this paper is organized as follows. Section II proposes the probabilistic classification vector machine algorithm, followed by experimental results and analysis in Section III. Section IV discusses the formulation of PCVMs by MAP analysis and margin analysis. Finally, Section V concludes the paper and presents some future work.

II. Probabilistic Classification Vector Machine

In this section, we will present the model specification for classification problems in Section II-A, then the prior over weight vectors will be discussed in Section II-B. Section II-C presents the detailed EM procedures for probabilistic classification vector machines.

A. Model Specification

Consider two-class classification and a data set of input–target training pairs $\{\mathbf{x}_i, y_i\}_{i=1}^N$, where $y_i \in \{-1, +1\}$. In order to map linear outputs to binary outputs, a link function should be chosen to allow a steep and smooth transition between two classes. This paper uses the probit link function

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

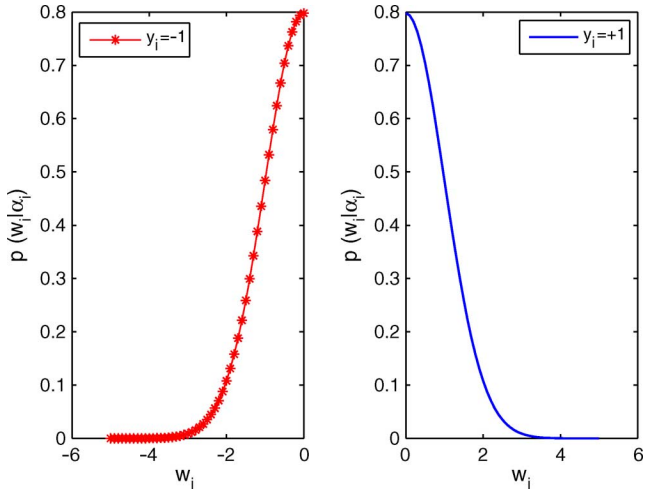


Fig. 3. Truncated Gaussian prior over weight vector $\mathbf{w}$. (a) When $y_i = -1$, $p(w_i|\alpha_i)$ is a nonpositive, right-truncated Gaussian prior. (b) When $y_i = +1$, $p(w_i|\alpha_i)$ is a nonnegative, left-truncated Gaussian prior.

where $\Psi(x)$ is the Gaussian cumulative distribution function. We use the probit link function because the probit link can be obtained from a simple latent variable model by the EM algorithm [19]. After incorporating the probit link function with the kernel method, the model becomes

$$l(\mathbf{x};\mathbf{w},b) = \Psi\left(\sum_{i=1}^{N} w_i \phi_{i,\theta}(\mathbf{x}) + b\right) = \Psi\left(\mathbf{\Phi}_\theta(\mathbf{x})\mathbf{w} + b\right). \quad (3)$$

B. Prior Over Weights

As discussed in Sections I, a truncated Gaussian prior is introduced for each weight w_i and a zero-mean Gaussian prior is adopted for the bias b

$$p(\mathbf{w}|\alpha) = \prod_{i=1}^{N} p(w_i|\alpha_i) = \prod_{i=1}^{N} N_t\left(w_i|0,\alpha_i^{-1}\right)$$
$$p(b|\beta) = N(b|0,\beta^{-1})$$

where b is the inverse variance of normal distribution, $N_t(w_i|0,\alpha_i^{-1})$ is a truncated Gaussian function, and α_i is the inverse variance. When $y_i = +1$, the truncated prior is a nonnegative, left-truncated Gaussian, and when $y_i = -1$, the prior is a nonpositive, right-truncated Gaussian. This can be formalized in (4) and illustrated in Fig. 3

$$p(w_i|\alpha_i) = \begin{cases} 2N\left(w_i|0,\alpha_i^{-1}\right), & \text{if } y_i w_i \geq 0 \\ 0, & \text{if } y_i w_i < 0. \end{cases} \quad (4)$$

In part A of the Appendix, we also discuss the model with hierarchical hyperpriors over α and β and present the probability $p(w_i)$ by incorporating the hyperpriors over α and β.

C. EM Algorithm

This section details the derivation of the EM algorithm. An EM algorithm [7] is a general algorithm for MAP estimation where the data are incomplete or the likelihood/prior function involves latent variables. EM iteratively alternates between performing an expectation (E) step and a maximization (M) step. In practice, derivation of equations in E and M steps needs to

be performed for different problems. In the following, we detail the model specification and the EM steps.

We follow the standard probabilistic formulation and assume that $\mathbf{\Phi}_\theta(\mathbf{x})\mathbf{w}+b$ is corrupted by an additive random noise ϵ, where $\epsilon \sim N(0,1)$. According to the probit link model, if $h_\theta(\mathbf{x}) = \mathbf{\Phi}_\theta(\mathbf{x})\mathbf{w}+b+\epsilon \geq 0$, $l=1$, and if $h_\theta(\mathbf{x}) = \mathbf{\Phi}_\theta(\mathbf{x})\mathbf{w}+b+\epsilon < 0$, $l=0$. We can obtain the probit mode as follows:

$$p(l=1|\mathbf{x},\mathbf{w},b) = p\left(\mathbf{\Phi}_\theta(\mathbf{x})\mathbf{w}+b+\epsilon \geq 0\right) = \Psi\left(\mathbf{\Phi}_\theta(\mathbf{x})\mathbf{w}+b\right). \tag{5}$$

$h_\theta(\mathbf{x})$ is a latent variable because ϵ is an unobservable variable. If the value of $h_\theta(\mathbf{x})$ were known, the likelihood of $\mathbf{w}$ could be given by the standard probabilistic formulation: $p(h_\theta(\mathbf{x})|\mathbf{w},b) = N(h_\theta(\mathbf{x})|\mathbf{\Phi}_\theta(\mathbf{x})\mathbf{w}+b,1)$. Consider the matrix $\mathbf{\Phi}_\theta = \left(\mathbf{\Phi}_\theta(\mathbf{x}_1)^T,\cdots,\mathbf{\Phi}_\theta(\mathbf{x}_N)^T\right)^T$, where $\mathbf{\Phi}_\theta(\mathbf{x}_i) = (\phi_\theta(\mathbf{x}_1,\mathbf{x}_i),\cdots,\phi_\theta(\mathbf{x}_N,\mathbf{x}_i))$ and vector $\mathbf{H}_\theta(\mathbf{x}) = (h_\theta(\mathbf{x}_1),\cdots,h_\theta(\mathbf{x}_N))^T$, then we obtain

$$p(\mathbf{H}_\theta|\mathbf{w},b) = (2\pi)^{-N/2}\exp\left\{-\frac{1}{2}\left\|\mathbf{H}_\theta - (\mathbf{\Phi}_\theta\mathbf{w}+bI)\right\|^2\right\}$$

where $I=(1,\cdots,1)^T$ is the N-dimension all-1 vector.

In order to obtain the complete log-posterior of $\mathbf{w}$ and b, α and β are also regarded as latent variables. Therefore, the latent variables in our formulation are: $\mathbf{H}_\theta(\mathbf{x}) = (h_\theta(\mathbf{x}_1),\cdots,h_\theta(\mathbf{x}_N))^T$, $\alpha = (\alpha_1,\cdots,\alpha_N)^T$, and the scalar β.

The log-posterior is given as follows:

$$\begin{aligned}
&\log p(\mathbf{w},b|\mathbf{y},\mathbf{H}_\theta,\alpha,\beta)\\
&\quad\propto \log p(\mathbf{H}_\theta|\mathbf{w},b)+\log p(\mathbf{w}|\alpha)+\log p(b|\beta)\\
&\quad\propto \mathbf{w}^T\mathbf{\Phi}_\theta^T(2\mathbf{H}_\theta-\mathbf{\Phi}_\theta\mathbf{w})+2bI^T\mathbf{H}_\theta-2bI^T\mathbf{\Phi}_\theta\mathbf{w}\\
&\qquad -b^2N-\mathbf{w}^T\mathbf{A}\mathbf{w}-\beta b^2
\end{aligned} \tag{6}$$

where $\mathbf{A}$ is a diagonal matrix $\mathbf{A}=\mathrm{diag}(\alpha_1,\cdots,\alpha_N)$.

1) Expectation Step: After obtaining the log-posterior, the expectation step, noted as a Q function, can be obtained by the following formula (refer to part B of the Appendix for detail):

$$\begin{aligned}
&Q(\mathbf{w},b|\mathbf{w}^{\mathrm{old}},b^{\mathrm{old}})\\
&\quad= E_{\mathbf{H}_\theta,\alpha,\beta}\left[\log p(\mathbf{w},b|\mathbf{y},\mathbf{H}_\theta,\alpha,\beta)|y,\mathbf{w}^{\mathrm{old}},b^{\mathrm{old}}\right]\\
&\quad= 2\mathbf{w}^T\mathbf{\Phi}_\theta^T\bar{\mathbf{H}}_\theta-\mathbf{w}^T\mathbf{\Phi}_\theta^T\mathbf{\Phi}_\theta\mathbf{w}+2bI^T\bar{\mathbf{H}}_\theta-2bI^T\mathbf{\Phi}_\theta\mathbf{w}\\
&\qquad -b^2N+\mathbf{w}^T\bar{A}\mathbf{w}-\bar{\beta}b^2
\end{aligned} \tag{7}$$

where $\bar{\mathbf{H}}_\theta = E[\mathbf{H}_\theta|y,\mathbf{w}^{\mathrm{old}},b^{\mathrm{old}}]$, $\bar{A}=\mathrm{diag}(E[\alpha_i|y_i,\mathbf{w}^{\mathrm{old}},b^{\mathrm{old}}])$, and $\bar{\beta}=E[\beta|y_i,\mathbf{w}^{\mathrm{old}},b^{\mathrm{old}}]$.

2) Maximization Step: In the maximization step, the partial derivatives with respect to $\mathbf{w}$, b, and each θ_k can be given by analyzing the derivative of (7)

$$\frac{\partial Q}{\partial \mathbf{w}} = -2\mathbf{\Phi}_\theta^T\mathbf{\Phi}_\theta\mathbf{w}+2\mathbf{\Phi}_\theta^T\bar{\mathbf{H}}_\theta-2b\mathbf{\Phi}_\theta^TI-2\bar{A}\mathbf{w} \tag{8}$$

$$\frac{\partial Q}{\partial b} = 2I^T\bar{\mathbf{H}}_\theta-2bN-2I^T\mathbf{\Phi}_\theta\mathbf{w}-2b\bar{\beta} \tag{9}$$

$$\frac{\partial Q}{\partial \theta_k} = 2\sum_{i=1}^{N}\sum_{j=1}^{N}\left\{(\mathbf{\Phi}_\theta\mathbf{w}-\bar{\mathbf{H}}_\theta)\mathbf{w}^T\odot\left(\frac{\partial\mathbf{\Phi}_\theta}{\partial\theta_k}\right)\right\}_{(i,j)} \tag{10}$$

where $\odot$ represents elementwise Hadamard matrix multiplication.

In general, the joint maximization of Q with respect to $\mathbf{w}$, b, and θ_k cannot be performed analytically. However, we can analytically obtain the optimal $\mathbf{w}$ and b by solving $\partial Q/\partial\mathbf{w}=0$ and $\partial Q/\partial b=0$, and then plug $\mathbf{w}^{\mathrm{new}}$ and b^{new} into Q. Maximization with respect to θ can be handled by any standard methods. This paper uses a simple conjugate gradient algorithm to obtain the optimal values of θ.

By setting $\partial Q/\partial\mathbf{w}=0$ and $\partial Q/\partial b=0$, the update rules of $\mathbf{w}$ and b can be analytically obtained

$$\mathbf{w}^{\mathrm{new}} = \left(\mathbf{\Phi}_\theta^T\mathbf{\Phi}_\theta+\bar{A}\right)^{-1}\left(\mathbf{\Phi}_\theta^T\bar{\mathbf{H}}_\theta-b\mathbf{\Phi}_\theta^TI\right) \tag{11}$$

$$b^{\mathrm{new}} = \frac{I^T\bar{\mathbf{H}}_\theta-I^T\mathbf{\Phi}_\theta\mathbf{w}}{\bar{\beta}+N}. \tag{12}$$

The pseudocode of PCVM can be summarized by Algorithm 1.

Algorithm 1: Probabilistic Classification Vector Machines

1: **Input**: $D=\{\mathbf{X},\mathbf{Y}\}=\{(x_n,y_n)\}_{n=1}^N$ is the training set; ker is the kernel type; θ is the kernel parameter; niter is the maximal iteration; $\mathbf{initVector}$ is the initialization vector; and threshold is the threshold value to determine whether the algorithm converges.
2: **Output**: The weight vector $\mathbf{w}$, bias b, and the updated kernel parameter θ.
3: $[\mathbf{w},b]=\mathrm{initialize}(\mathbf{initVector})$;
4: $\mathbf{nonZero}=\mathrm{determine\_nonZero\_Vector}(\mathbf{w})$;
5: **for** $i=1$ to niter **do**
6: $\quad\mathbf{\Phi}=\mathrm{Kernel\_Matrix\_Calculation}(\mathbf{X},\mathbf{Y},ker,\theta)$;
7: $\quad\mathbf{w}^{\mathrm{new}}=\mathrm{weight\_update}(\mathbf{\Phi},\mathbf{w},\mathbf{Y},\mathbf{nonZero})$;
8: $\quad b^{\mathrm{new}}=\mathrm{bias\_update}(\mathbf{\Phi},b,\mathbf{Y},\mathbf{nonZero})$;
9: $\quad\theta^{\mathrm{new}}=\mathrm{parameter\_update}(\mathbf{\Phi},\mathbf{X},\mathbf{Y},\mathrm{ker},\theta,\mathbf{w}^{\mathrm{new}},b^{\mathrm{new}},\mathbf{nonZero})$;
10: $\quad\mathbf{nonZero}^{\mathrm{new}}=\mathrm{determine\_nonZero\_Vector}(\mathbf{w}^{\mathrm{new}})$;
11: $\quad$**if** $\max(\mathrm{abs}(\mathbf{w}^{\mathrm{new}}-\mathbf{w}))<\mathrm{threshold}$ **then**
12: $\quad\quad$break;
13: $\quad$**else**
14: $\quad\quad$continue;
15: $\quad$**end if**
16: **end for**

Algorithm 1 includes the following major steps.

1) Initialize the weight vector $\mathbf{w}$ with an initialization vector and generate an indicator vector $\mathbf{nonZero}$ to indicate which elements are nonzero in $\mathbf{w}$ (lines 3 and 4).
2) Compute the kernel matrix $\mathbf{\Phi}$ (line 6).
3) Update the weight vector $\mathbf{w}$ according to (11) (line 7).
4) Update the bias b according to (12) (line 8).
5) Update the kernel parameter θ according to (10) (line 9).
6) Update the indicator vector $\mathbf{nonZero}$ (line 10).
7) Compare the new and old weight vectors $\mathbf{w}^{\mathrm{new}}$ and $\mathbf{w}$ to see whether the algorithm converges. If so, terminate the algorithm. Otherwise, jump to step (b) and continue the loop (lines 11–15).

In the above algorithm, to avoid numerical singularity, we use an indicator vector to indicate which elements of the weight vector $\mathbf{w}$ are to be set to zero[4] and prune the corresponding columns of $\mathbf{\Phi}$. As explained by Tipping [26, App. B.1, p. 235], even though in theory the matrix $(\mathbf{\Phi}_\theta^T\mathbf{\Phi}_\theta + \bar{\mathbf{A}})$ is positive definite, it may become numerically singular when some of the diagonal elements in matrix $\bar{\mathbf{A}}$ tend towards large values. In this case, we thus prune the corresponding basis function from the model at that point (i.e., by deleting the appropriate column from $\mathbf{\Phi}$) to avoid ill-conditioning. Such a procedure of pruning basis functions has also been adopted, e.g., in [12]. More details can be found in part C of the Appendix.

Part C of the Appendix presents some minor modifications to $\mathbf{w}^{\text{new}}$ and b^{new} for a stable numerical computation in practice.

III. Experimental Studies

First, we present experimental results of PCVMs, SVMs, and RVMs on four synthetic data sets in order to understand the behaviors of these algorithms. Second, we carry out extensive experiments on 13 benchmark data sets using three performance metrics: the error rate (ERR), the area under the curve of receiver operating characteristic (AUC), and the root mean squared error (RMSE). Finally, we present detailed statistical tests including five replications of twofold cross-validation F test, i.e., 5×2 cv F test [1], over the single data set and Friedman test [14] with the corresponding post-hoc tests over multiple data sets for multiple classifiers.

A. Synthetic Data Sets

In the first experiment, we compare PCVMs, soft-margin SVMs [3], and RVMs [26] on four 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. *Bumpy* comes from two equal Gaussians but being rotated by 90°, and quadratic boundaries are required. *Relevance* represents a case where only one dimension of the data is relevant to separating the data.

This experiment employs a Gaussian RBF kernel as the basis function

$$\phi_\theta(\mathbf{x}, \mathbf{x}_i) = \exp\left\{-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{\theta^2}\right\} \tag{13}$$

where θ is the width of a Gaussian kernel.

The parameters of SVMs including the regularization parameter C and the kernel parameter θ are selected by grid search with tenfold cross validation.[5] The kernel parameter θ of RVMs is selected by tenfold cross validation.

Although PCVMs could optimize the kernel parameter by maximizing the expectation, the EM algorithm is sensitive to the initialization point and might get stuck in local maxima. In order to avoid the local maxima, we choose different initialization points to run multiple times and choose the best one using cross validation. This model selection procedure is carried out by training each data set with five different initial values of θ. From the resulting solutions (five per data set), we select the initialization point that produces minimal test errors.

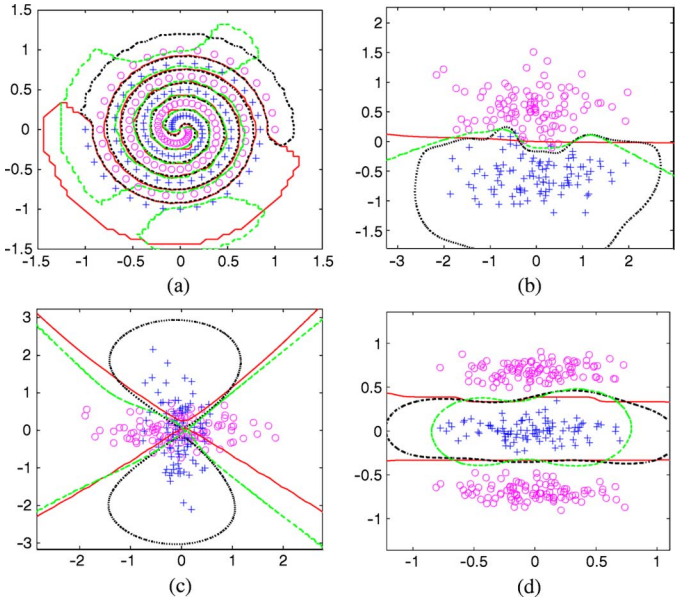


Fig. 4. Comparison of classification of synthetic data sets using an RBF kernel. Two classes are shown as pluses and circles. The separating lines are obtained by projecting test data over a grid. Dark, dashed, and dotted–dashed lines are obtained by PCVMs, RVMs, and SVMs, respectively. Kernel and regularization parameters for SVMs and RVMs are obtained by tenfold cross validation, whereas the parameters of PCVMs are obtained by an EM algorithm. (a) Spiral. (b) Overlap. (c) Bumpy. (d) Relevance.

In Fig. 4, we present the decision boundaries of three algorithms. We can observe a similar performance of PCVMs and SVMs in the case of *Spiral*. RVMs cannot obtain the correct decision boundary due to the highly nonlinear data set. The failure indicates that the prior of RVMs produces excessive sparseness in the outer part of data, leading the boundary biasing towards outer circle and hence producing errors. PCVMs perform well because they generate adequate sparseness in both inner and outer circles from the truncated prior.

It is encouraging to observe that PCVMs give more accurate results in the rest of the cases. PCVMs produce almost linear decision boundary in *Overlap* and RVMs give analogously curving decision boundary, whereas SVMs overfit.[6] In *Bumpy*, PCVMs and RVMs give similar quadratic solutions, with PCVMs having the smoothest boundary and SVMs having the localized boundary. Finally, all the algorithms provide accurate results for *Relevance*, with PCVMs giving the smoothest solution.

The results of PCVMs are promising on these four synthetic data sets. PCVMs not only handle the data sets with a predominating linear or quadratic decision boundary, e.g., *Overlap*

[4]The elements w_i of $\mathbf{w}$ whose corresponding values of α_i become large.

[5]The ranges of cross-validation search for SVM are $C \in \{1, 2, \cdots, 100\}$ 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 RVM in both synthetic data sets and benchmark data sets.

[6]With a large radius parameter, SVM can generate a linear boundary. However, due to the small data size, tenfold cross validation selects a smaller radius with smaller CV error.

and *Bumpy*, but also being applied to the highly nonlinear data sets, e.g., *Spiral* and the data sets with redundant features, e.g., *Relevance*.

B. Benchmark Data Sets

In order to evaluate the performance of PCVMs further, we compare different algorithms on 13 well-known benchmark problems. These algorithms are soft-margin SVMs (SVM_{soft}) [3], hard-margin SVMs (SVM_{hard}) [3], SVMs whose kernel parameters are optimized by PCVMs (SVM_{PCVM}), relevance vector machines (RVMs) [26], and PCVMs. We report the algorithm SVM_{PCVM} since it provides the opportunity to test whether the kernel parameter, optimized by PCVMs, works for SVMs as well. This 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), one-nearest neighbor (1NN) and k-nearest neighbor (kNN), where the number of nearest neighbors k is selected by the parameter selection methodology (where k is selected from $\{1, 2, \cdots, 20\}$).

This paper uses the data sets, which have been preprocessed and organized by Rätsch *et al.*[7] to do binary classification tests. These data sets include one synthetic set (Banana) along with 12 other real-world data sets from the University of California at Irvine (UCI) [20] and DELVE.[8] The characteristics of the data set are summarized in Table II.

The main difference between the original and Rätsch's data is that Rätsch converted every problem into binary classes and randomly partitioned every data set into 100 training and testing instances (Splice and Image have only 20 splits in the Rätsch's implementation and we generate additional 80 splits by random sampling to make our experiments consistent). In addition, every instance was input-normalized dimensionwise to have zero mean and unit standard deviation.

The ERR, the AUC, and the RMSE represent three most often used metrics, which represent threshold metric, probability metric, and rank metric, respectively [5]. In our paper, we will use the three performance metrics for binary classification problems.

The procedure of parameter optimization follows Rätsch's methodology [23], 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_{soft}, we train soft-margin SVMs 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 SVM_{hard}, SVM_{PCVM}, RVMs, and kNN. The only difference among them is that they need to optimize different parameters. For SVM_{hard} and RVMs, we need to optimize the kernel width parameter θ. SVM_{PCVM} adopts the optimized kernel parameter by PCVMs and so it only needs to optimize C. For kNN, the number of nearest neighbors is selected by this methodology as well.

The PCVM has only one parameter θ, which can be automatically optimized in the training process. However, as we know, the EM algorithm is prone to converge to local maxima. The usual approach to avoid the local maxima is to run the EM algorithm multiple times from different initialization points and choose the best one based on cross-validation error rate.

To select the best initialization point of PCVMs, we try to follow the same procedure. We train a PCVMs model with different initializations (eight initializations[9] in this paper) over the first five training folds of each data set. Hence, 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 five elements. Then, we select the median over those parameters.

Table I reports the performance of these algorithms on the 13 benchmark data sets with ERR, AUC, and 1-RMSE. According to that table, the PCVM performs very well in terms of three different metrics. For example, under the ERR metric, it is observed that the PCVM outperforms all other methods in six out of 13 data sets, comes second in three cases, and third in the remaining four. The PCVM performs extremely well under the AUC metric, with the first place in ten cases and the second in the remaining three. Even when the PCVM fails under other metrics on one of the data sets, e.g., Cancer or Titanic, 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 suboptimal results.

The experimental results for the three variants of SVMs are also enlightening.

The soft-margin SVM is consistently better than the hard-margin SVM under the ERR and AUC metrics. Under the RMSE metric, the hard-margin SVM is slightly better than (or almost as good as) the soft-margin SVM on two data sets: Image and Thyroid.

In most cases, the SVM_{PCVM} is worse than or comparable to the corresponding PCVM; it achieves similar or better performance than the soft-margin SVM. This indicates that the optimized kernel parameter by the PCVM works well for the SVM. Our results indicate that the PCVM procedure performs better than cross validation, even when it comes to fitting the SVM kernel parameters.

The baseline algorithms, 1NN, kNN, and LDA/QDA, only perform well on one or two data sets. In all other cases, they fail to compete with PCVM and SVMs, especially under the AUC metric.

Another interesting point is that the PCVM achieves better performance by employing only a few of the data points, which is illustrated in Table III.

According to Table III, the number of support vectors for SVM grows almost linearly with the number of training points, while the RVM consistently uses much fewer data points. The

[7]http://www.ida.first.fraunhofer.de/projects/bench/benchmarks.htm

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

[9]The RVM and SVM solutions are supplied as two initialization, in which the zero weights and reverse signed weights in RVM are replaced with small random values to avoid being pruned in the first learning step. The other six initializations are performed randomly.

TABLE I
COMPARISON OF 1NN, kNN, LDA, QDA, SVM_{soft}, SVM_{hard}, SVM_{PCVM}, RVM, AND PCVM ON 13 BENCHMARK DATA SETS, BY PERCENT ERROR, AUC, (1-RMSE), AND (STANDARD DEVIATION). THESE RESULTS ARE THE AVERAGE OF 100 RUNS ON THE DATA SETS

Error	Banana	Cancer	Diabetics	Solar	German	Heart	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
1NN	13.64(0.76)	32.70(4.84)	30.12(2.05)	39.22(5.05)	29.46(2.47)	23.16(3.74)	2.82(0.59)	35.03(1.36)	27.64(1.47)	**4.36(2.21)**	33.00(11.92)	6.68(0.72)	15.83(0.65)
kNN	11.26(0.54)	30.58(4.19)	25.83(1.90)	35.74(2.20)	25.44(2.52)	16.46(3.61)	2.82(0.59)	35.03(1.36)	24.00(2.35)	**4.36(2.21)**	23.84(4.94)	2.81(0.20)	10.98(0.67)
LDA	46.69(4.81)	31.81(4.35)	24.66(2.03)	34.41(1.75)	28.52(2.56)	**16.40(2.87)**	16.89(0.88)	24.62(0.66)	16.26(0.53)	12.85(3.30)	23.57(4.61)	2.61(0.17)	17.66(0.60)
QDA	39.58(3.38)	31.36(4.91)	26.88(1.96)	44.74(1.82)	30.01(2.79)	19.64(3.24)	17.43(1.97)	2.58(0.32)	14.77(0.94)	6.85(2.37)	24.30(6.43)	3.45(0.29)	16.65(0.83)
SVM_{soft}	11.56(0.68)	26.38(4.76)	24.34(1.73)	**32.54(1.75)**	24.14(2.18)	17.82(3.27)	2.78(0.56)	1.66(0.12)	10.70(0.63)	4.79(2.04)	22.69(0.86)	2.69(0.15)	**10.25(0.43)**
SVM_{hard}	11.98(0.71)	28.97(4.87)	30.68(2.28)	35.98(3.08)	26.63(2.35)	22.29(3.50)	2.95(0.58)	1.66(0.12)	10.70(0.63)	5.04(2.14)	**22.30(1.05)**	2.93(0.27)	10.95(0.57)
SVM_{PCVM}	10.47(0.49)	**25.60(4.45)**	23.82(1.73)	32.65(1.69)	24.73(2.29)	17.49(3.26)	2.75(0.57)	1.64(0.12)	**10.51(0.62)**	4.79(2.24)	22.54(0.95)	2.84(0.17)	10.80(0.52)
RVM	10.78(0.52)	26.60(4.70)	23.81(1.84)	35.29(1.87)	24.52(2.32)	17.30(3.56)	3.15(0.66)	2.15(0.64)	12.94(0.71)	5.12(2.62)	23.30(1.50)	3.32(0.43)	10.80(0.64)
PCVM	**10.30(0.76)**	26.23(4.62)	**23.15(1.95)**	32.89(1.79)	**23.62(2.24)**	16.62(3.45)	**2.49(0.52)**	**1.53(0.12)**	10.60(0.65)	4.55(2.49)	22.58(1.37)	**2.46(0.26)**	10.40(0.58)
AUC	Banana	Cancer	Diabetics	Solar	German	Heart	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
1NN	86.24(0.77)	60.37(5.58)	66.08(2.24)	60.26(5.90)	63.71(2.92)	76.67(3.79)	96.97(0.62)	64.63(1.30)	73.19(1.40)	94.14(3.44)	63.18(6.47)	93.32(0.72)	82.29(1.05)
kNN	88.37(0.53)	59.82(4.71)	68.00(2.16)	64.30(2.63)	62.88(2.83)	83.03(3.67)	96.97(0.62)	64.63(1.30)	77.31(2.08)	94.14(3.44)	67.89(3.51)	97.19(0.20)	88.63(1.08)
LDA	53.42(4.88)	66.19(4.90)	74.00(2.18)	66.53(1.58)	71.37(2.57)	83.26(2.94)	81.80(0.88)	75.30(0.65)	84.00(0.51)	79.37(4.99)	70.08(2.36)	97.39(0.17)	86.20(0.58)
QDA	60.88(3.31)	63.69(5.52)	71.14(2.30)	50.00(0.00)	69.38(2.58)	80.14(3.33)	82.70(1.76)	97.43(0.31)	84.59(0.94)	89.97(3.56)	69.62(4.13)	96.55(0.29)	81.72(1.52)
SVM_{soft}	95.09(0.57)	68.95(5.69)	82.42(1.75)	71.87(2.34)	77.72(2.57)	89.25(2.97)	99.30(0.25)	99.84(0.01)	95.93(0.31)	99.01(0.74)	70.78(2.87)	99.51(0.04)	96.22(0.25)
SVM_{hard}	94.44(0.69)	66.15(6.56)	73.08(1.98)	66.94(3.08)	73.62(2.72)	85.85(2.92)	98.92(0.33)	99.84(0.01)	95.93(0.31)	98.81(0.89)	70.34(3.03)	99.45(0.07)	95.62(0.35)
SVM_{PCVM}	96.14(0.30)	69.93(5.77)	83.04(1.68)	71.95(2.54)	77.37(2.57)	89.50(2.87)	99.27(0.26)	99.85(0.01)	**96.12(0.31)**	**99.41(0.54)**	71.20(2.43)	99.47(0.05)	95.74(0.31)
RVM	95.97(0.38)	71.01(5.68)	82.69(1.59)	**72.75(1.73)**	78.41(2.73)	89.52(2.87)	97.78(0.41)	99.31(0.02	94.04(0.48)	98.49(1.71)	72.98(2.49)	93.14(1.31)	96.04(0.43)
PCVM	**96.30(0.32)**	**72.98(6.44)**	**85.01(2.10)**	72.46(1.83)	**79.93(2.80)**	**91.08(2.84)**	**99.63(0.31)**	**99.86(0.01)**	96.04(0.32)	99.21(0.89)	**75.42(1.28)**	**99.78(0.03)**	**96.42(0.31)**
1-RMSE	Banana	Cancer	Diabetics	Solar	German	Heart	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
1NN	63.08(1.03)	42.97(4.19)	45.15(1.85)	37.50(3.95)	45.77(2.29)	52.04(3.95)	**83.30(1.73)**	40.83(1.15)	47.44(1.40)	79.84(5.45)	43.41(9.92)	74.18(1.36)	60.22(0.82)
kNN	66.46(0.80)	44.83(3.82)	49.21(1.89)	40.24(1.82)	49.62(2.52)	59.67(4.46)	**83.30(1.73)**	40.83(1.15)	51.07(2.39)	79.84(5.45)	51.36(4.35)	83.26(0.59)	66.88(1.00)
LDA	31.75(3.47)	43.74(3.87)	50.38(2.05)	41.36(1.49)	46.65(2.38)	59.66(3.59)	58.91(1.08)	50.38(0.66)	59.68(0.66)	64.47(4.83)	51.59(3.67)	83.85(0.53)	57.99(0.71)
QDA	37.15(2.71)	44.17(4.43)	48.19(1.89)	33.13(1.36)	45.28(2.54)	55.84(3.73)	58.32(2.37)	**83.97(0.98)**	61.59(1.22)	74.25(4.76)	50.96(5.03)	81.45(0.79)	59.21(1.01)
SVM_{soft}	70.50(0.85)	56.52(3.23)	59.54(1.09)	51.14(1.48)	59.25(1.27)	63.49(2.21)	80.42(0.72)	80.88(0.38)	**69.93(0.46)**	78.49(2.49)	**58.16(0.73)**	75.51(0.31)	70.44(0.31)
SVM_{hard}	69.64(0.90)	52.46(3.90)	55.35(0.88)	48.46(1.23)	57.20(1.34)	60.91(2.08)	80.43(0.79)	80.88(0.38)	**69.93(0.46)**	78.51(2.57)	57.99(0.84)	75.31(0.51)	70.02(0.46)
SVM_{PCVM}	**72.33(0.49)**	**57.31(3.09)**	**59.88(1.11)**	51.14(1.43)	58.99(1.22)	63.93(2.31)	80.80(0.76)	79.77(0.31)	67.84(0.41)	81.42(2.79)	58.00(0.60)	75.42(0.36)	70.15(0.42)
RVM	71.24(0.88)	55.95(3.66)	58.54(1.37)	**53.70(1.08)**	57.73(1.94)	62.65(3.76)	79.69(1.29)	79.88(1.87)	66.98(0.97)	80.10(5.74)	57.61(1.42)	83.14(1.31)	70.06(1.18)
PCVM	72.13(0.71)	53.30(4.93)	59.41(1.84)	51.64(1.16)	**59.89(1.83)**	**64.73(3.82)**	81.80(0.77)	80.36(0.62)	68.97(0.56)	**81.85(4.29)**	56.42(1.58)	**85.02(0.72)**	**71.78(0.82)**

TABLE II
SUMMARY OF 13 BENCHMARK DATA SETS

Data	No. Train	No. Test	Positive %	Negative %	Dim
Banana	400	4900	44.83%	55.17%	2
Cancer	200	77	29.28%	70.72%	9
Diabetics	468	300	34.90%	65.10%	8
Solar	666	400	65.28%	34.72%	9
German	700	300	30.00%	70.00%	20
Heart	170	100	44.44%	55.56%	13
Image	1300	1010	56.95%	43.05%	18
Ringnorm	400	7000	49.51%	50.49%	20
Splice	1000	2175	44.93%	55.07%	60
Thyroid	140	75	30.23%	69.77%	5
Titanic	150	2051	58.33%	41.67%	3
Twonorm	400	7000	50.04%	49.96%	20
Waveform	400	4600	32.94%	67.06%	21

PCVM employs more vectors than the RVM but much fewer than SVM. This observation goes in accordance with the formulation. In the RVM, the weights could reach zero from both sides because of the symmetrical zero-mean Gaussian, whereas the weights in PCVMs could only converge to zero from one side because of the truncated Gaussian prior. It is worth noting that the PCVM has better performance than the RVM according to Table I.

C. Statistical Comparisons on Single Data Sets

In order to compare the PCVM with other algorithms in a sound statistical context, we perform the statistical test for paired classifiers, e.g., PCVM versus SVM_{soft} and PCVM versus RVM, on each single data set. We will carry out statistical tests on these three metrics and provide the win–loss–tie summary for these metrics. The threshold of the statistical tests is set to be 0.05.

Although t test has been used in most of the literatures to conduct statistical tests, it has been criticized for its type I/II error and low power for a long time [9]. Dietterich [9] analyzes five statistical tests and proposes a new test, five replications of twofold cross-validation t test, i.e., 5×2 cv t test, which has a low type I error and a reasonable power [9].

However, 5×2 cv t test takes the statistics from only one fold as the numerator and may vary depending on factors that should not affect the test. Alpaydin [1] improved 5×2 cv t test by combining multiple statistics to get a more robust test, 5×2 cv F test, which has a lower type I error and a higher power. In this paper, we compare algorithms using 5×2 cv F test [1].

In the 5×2 cv F test, five replications of twofold cross-validation have been conducted. In each replication, the data set is divided into two equal-sized sets. $p_i^{(j)}$ is the difference between the error rates of the two classifiers on fold $j = 1, 2$ of replication $i = 1, \ldots, 5$. The average on replication i is $\bar{p}_i = (p_i^{(1)} + p_i^{(2)})/2$, and the estimated variance is $s_i^2 = (p_i^{(1)} - \bar{p}_i)^2 + (p_i^{(2)} - \bar{p}_i)^2$.

The 5×2 cv F test combines the results of the ten statistics $p_i^{(j)}$ as the numerator, which makes the test more robust. Alpaydin [1] pointed out that the following statistics:

$$f = \frac{\sum_{i=1}^{5} \sum_{j=1}^{2} \left(p_i^{(j)}\right)^2}{2 \sum_{i=1}^{5} s_i^2} \tag{14}$$

is approximately F distributed with ten and five degrees of freedom, $F(10, 5)$, and used this statistics to conduct the 5×2 cv F test.

Table IV gives the win–loss–tie summary of the 5×2 cv F test based on 13 benchmark data sets. The significance tests show that SVM_{PCVM} is close to the PCVM under the RMSE metric; and SVM_{PCVM} wins three times and loses four times. This situation occurs for SVM_{soft} as well. SVM_{soft} wins three times and loses five times under RMSE.

However, under the other two metrics, the differences between $\text{SVM}_{\text{soft}}/\text{SVM}_{\text{PCVM}}$ and the PCVM are greater. 1)

TABLE III
COMPARISON OF $\mathrm{SVM}_{\mathrm{soft}}$, $\mathrm{SVM}_{\mathrm{hard}}$, $\mathrm{SVM}_{\mathrm{PCVM}}$, RVM, AND PCVM ON 13 BENCHMARK DATA SETS, BY NUMBER OF VECTORS AND STANDARD DEVIATION. THESE RESULTS ARE THE AVERAGE OF 100 RUNS ON THE DATA SETS

Vectors	Train	$\mathrm{SVM}_{\mathrm{soft}}$	$\mathrm{SVM}_{\mathrm{hard}}$	$\mathrm{SVM}_{\mathrm{PCVM}}$	RVM	PCVM
Banana	400	89.8±10.4	88.3±11.1	111.1±8.9	12.7±1.5	19.8±4.3
Cancer	200	119.9±6.4	105.7±6.4	125.2±6.25	9.4±2.0	9.4±2.6
Diabetics	468	270.6±7.9	376.0±7.0	265.0±7.0	8.4±2.0	19.8±3.1
Solar	666	595.3±15.6	613.2±13.7	593.3±15.5	21.5±2.4	41.8±8.4
German	700	520.9±10.9	514.1±12.4	547.7±9.9	27.4±4.3	40.9±7.6
Heart	170	111.7±4.8	104.8±6.1	103.7±5.3	9.3±1.8	5.9±2.2
Image	1300	422.4±11.3	397.1±11.0	396.9±11.4	123.8±9.6	170.3±11.7
Ringnorm	400	151.2±9.0	151.2±8.9	198.6±10.8	6.5±1.7	17.6±3.8
Splice	1000	594.7±16.5	594.8±16.0	778.8±13.6	89.0±6.0	123.2±8.3
Thyroid	140	53.1±3.2	51.5±3.7	24.8±3.7	4.1±0.8	8.6±2.3
Titanic	150	147.5±4.7	143.5±6.3	142.0±9.6	6.8±1.4	16.6±2.4
Twonorm	400	237.4±7.1	236.9±8.4	235.9±7.7	7.7±1.1	13.8±3.4
Waveform	400	229.6±8.8	225.3±9.2	224.9±9.1	20.7±3.1	26.3±3.7

TABLE IV
5×2 cv F TEST FOR 13 DATA SETS. FOR EACH METRIC, THE FIRST LINE IS THE WIN–LOSS–TIE SUMMARY OF THE ALGORITHM AGAINST THE PCVM BASED ON THE MEAN VALUE. THE SECOND ROW GIVES THE STATISTICAL SIGNIFICANCE WIN–LOSS–TIE SUMMARY BASED ON 13 BENCHMARK DATA SETS

Data Sets	1NN	kNN	LDA	QDA	$\mathrm{SVM}_{\mathrm{soft}}$	$\mathrm{SVM}_{\mathrm{hard}}$	$\mathrm{SVM}_{\mathrm{PCVM}}$	RVM
ERR Mean	1-12-0	2-11-0	1-12-0	0-13-0	2-11-0	0-13-0	3-10-0	0-13-0
Significant	1-12-0	2-9-2	1-10-2	0-13-0	1-8-4	0-10-3	2-7-4	0-10-3
AUC Mean	0-13-0	0-13-0	0-13-0	0-13-0	0-13-0	0-13-0	1-12-0	1-12-0
Significant	0-13-0	0-13-0	0-13-0	0-13-0	0-9-4	0-10-3	0-7-6	1-8-4
RMSE Mean	1-12-0	1-12-0	0-13-0	1-12-0	5-8-0	3-10-0	4-9-0	3-10-0
Significant	1-11-1	1-10-2	0-13-0	1-12-0	3-5-5	2-9-2	3-4-6	3-7-3

$\mathrm{SVM}_{\mathrm{PCVM}}$ wins two times and loses seven times under ERR and never wins under AUC. 2) $\mathrm{SVM}_{\mathrm{soft}}$ wins once and loses eight times under ERR and never wins under AUC. The RVM does not seem to perform well under the ERR metric since it never wins. Under other metrics, RVM seems to be comparable to the $\mathrm{SVM}_{\mathrm{soft}}$.

The performance of $\mathrm{SVM}_{\mathrm{hard}}$ is not competitive against the PCVM. It only wins twice under the RMSE metric. The experimental results also reveal that these baseline algorithms underperform significantly against other algorithms.

This section has presented the statistical tests over single data sets. The next section will present the statistical comparisons over multiple data sets and analyze the reasons why the PCVM performs better than other algorithms.

D. Statistical Comparisons Over Multiple Data Sets

In the previous section, we have conducted the statistical tests on single data sets. It is difficult to statistically compare these algorithms based on multiple data sets, since the differences among these classifiers are significant for some data sets but not for other data sets.

In general, counting the number of times an algorithm performs better, worse, or equal to the others is a common approach. Some authors prefer to count only significant wins and losses, where the significance is determined using a statistical test on each data set, for example, Dietterich's 5×2 cv t test [9]. However, this statement is not reliable since it puts an arbitrary threshold of 0.05 or 0.10 on what counts and what does not for each data set. This can be shown by a simple scenario [8].

Suppose that we compare two algorithms on 1000 different data sets. In each and every case, algorithm A is better than algorithm B, but the difference is never significant. It is true that for each single case, the difference between the two algorithms can be attributed to a random chance, but how likely is it that one algorithm is just lucky in all 1000 out of 1000 independent experiments?

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 [13], [14] with the corresponding post-hoc tests. The Friedman test is a nonparametric equivalence 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 [15].

The Friedman test is carried out to test whether all the algorithms are equivalent. If the test result rejecting 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 PCVM as the control classifier to be compared with. Since the baseline classification algorithms are not comparable to SVMs, RVMs, and PCVMs, this section will analyze only four algorithms: $\mathrm{SVM}_{\mathrm{soft}}$, $\mathrm{SVM}_{\mathrm{hard}}$, $\mathrm{SVM}_{\mathrm{PCVM}}$, and RVMs against the control classifier PCVM.

The Bonferroni–Dunn test [10] 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

TABLE V
THE MEAN RANK OF THESE ALGORITHMS UNDER THE THREE METRICS: ERR, AUC, AND RMSE

Rank	1NN	kNN	LDA	QDA	SVM_{soft}	SVM_{hard}	SVM_{PCVM}	RVM	PCVM
ERR	7.5	5.3	6.3	7.7	3.4	5.5	2.9	4.7	1.9
AUC	8.0	7.4	6.7	7.5	3.3	4.7	2.6	3.6	1.2
RMSE	7.6	5.9	6.9	7.3	3.3	4.7	3.0	3.9	2.4

TABLE VI
FRIEDMAN TESTS WITH THE CORRESPONDING POST-HOC TESTS, BONFERRONI–DUNN, TO COMPARE CLASSIFIERS FOR MULTIPLE DATA SETS. THE THRESHOLD IS 0.10, AND $q_{0.10} = 2.241$

Metrics	Friedman test	$\text{CD}_{0.10}$	SVM_{soft}	SVM_{hard}	SVM_{PCVM}	RVM
ERR	0.00	1.38	1.35	2.69	0.85	2.42
AUC	0.00	1.38	2.08	3.31	1.38	2.08
1-RMSE	0.01	1.38	0.50	1.92	0.42	1.38

corresponding average ranks[10] differ by at least the critical difference

$$CD = q_\alpha \sqrt{\frac{j(j+1)}{6T}} \tag{15}$$

where j is the number of algorithms, T is the number of data sets, and critical values q_α can be found in [8]. For example, when $j = 5$, $q_{0.10} = 2.241$, where the subscript 0.10 is the threshold value.

Table V lists the mean rank of these algorithms under the three metrics: ERR, AUC, and 1-RMSE. Table VI gives the Friedman test results. Since we employ the same threshold 0.10 for all three metrics, the critical difference $CD = 1.38$, where $j = 5$ 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 PCVM versus SVM_{hard} and PCVM versus RVM are greater than the critical difference, so the differences are significant, which means the PCVM is significantly better than SVM_{hard} and RVM in this case. The difference between PCVM and SVM_{soft} is just below the critical difference, which seems to suggest that SVM_{soft} is likely to be different from PCVM. We could not detect any significant difference between SVM_{PCVM} and PCVM. The correct statistical statement would be that *the experimental data are not sufficient to reach any conclusion regarding the difference between PCVM and* SVM_{PCVM}.

Second, the PCVM significantly outperforms all other algorithms under the AUC metric. Since AUC metric requires relative accurate scores to discriminate positive and negative instances [11], PCVMs succeed by generating the probabilistic outputs. Another reason is that AUC is insensitive to the class skew/distribution [11] and some data sets used in this paper are imbalanced. In this way, PCVMs perform well on these skewed data sets by considering different priors for different classes and thus have better scores under the AUC metric.

Third, under the RMSE metric, only the differences between PCVM and SVM_{hard}/RVM are significant. Since the differences between PCVM and $\text{SVM}_{\text{soft}}/\text{SVM}_{\text{PCVM}}$ are smaller than the critical difference, we cannot draw any conclusion about the difference between PCVM versus SVM_{soft} and PCVM versus SVM_{PCVM} under the RMSE metric in our experimental settings.

There are three major reasons why the PCVM performs better than others.

1) PCVM generates adequate robustness and sparseness because of 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 skewness, the PCVM provides probabilistic outputs to assess the uncertainty for the predictions and performs well on these skewed data sets, which explains why the PCVM is so good under the AUC metric. Although the RVM also provides probabilistic outputs, it adopts an improper prior over weights and thus leads to inferior results.
3) The PCVM incorporates an efficient parameter optimization procedure based on probabilistic inference and the EM algorithm. This procedure not only saves the effort to do cross-validation grid search but also improves the performance.

E. Algorithm Complexity

Both classical SVMs algorithms and PCVMs have a time complexity of $O(N^3)$, where N is the number of training points, but the computational complexity of SVMs can be reduced to approximately $O(N^{2.1})$ for sequential minimal optimization (SMO)-like algorithms [16], which breaks the large quadratic programming (QP) problem into a series of smallest possible QP problems.

In PCVMs, 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 *nonzero* basis functions and $M \leq N$.

This computational complexity leads to longer training times and larger memory usage. However, because of the sparseness-inducing prior and quick convergence of the EM algorithm,

[10]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. Refer to Table V for the mean rank of these algorithms under different metrics.

TABLE VII
RUNNING TIME OF THE PCVM, SVM_{soft}, RVM, LDA, QDA, 1NN, AND kNN ON 13 DATA SETS IN SECONDS. RESULTS ARE AVERAGED OVER 100 RUNS

Time(s)	Banana	Cancer	Diabetics	Solar	German	Heart	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
PCVM	6.71	1.14	10.30	8.93	14.2	1.05	92.52	9.37	70.41	0.30	1.25	13.21	11.47
SVM_{soft}	5.41	0.51	3.17	2.86	3.81	0.14	12.09	6.98	15.85	0.09	0.29	7.58	4.01
RVM	7.76	2.39	14.32	12.39	28.96	1.74	209.84	9.34	95.42	0.68	2.19	14.29	18.18
LDA	0.04	0.02	0.03	0.03	0.03	0.02	0.04	0.09	0.10	0.09	0.02	0.07	0.06
QDA	0.03	0.02	0.02	0.03	0.02	0.02	0.04	0.07	0.09	0.09	0.02	0.07	0.06
1NN	0.70	0.04	0.10	0.21	0.21	0.04	1.35	4.45	9.59	0.02	0.10	3.10	2.04
kNN	26.23	0.68	2.91	6.26	6.21	0.71	40.63	96.90	213.36	0.37	4.52	89.58	66.89

TABLE VIII
THE MEAN RANK OF COMPUTATIONAL TIME

Rank	PCVM	SVM_{soft}	RVM	LDA	QDA	1NN	kNN
Time	5.5	4	6.5	1.8	1.4	2.8	5.9

PCVMs prune the basis functions rapidly from $M = N$ at initialization to a small size for most problems. Also, this disadvantage of PCVMs is offset by the lack of need to perform cross validation over parameters, such as C and kernel parameter θ in SVMs.

Table VII shows the average running time of PCVMs, SVM_{soft},[11] RVMs, LDA, QDA, 1NN, kNN on 13 data sets in seconds. Results are averaged over 100 runs. Note that in Table VII, we do not record the cross-validation time for SVM_{soft} and RMVs, but the running time of kNN includes the time to perform tenfold cross validation ($k \in \{1, \cdots, 20\}$). We rank these algorithms based on the computational time on each data set and record the ranking of each algorithm as 1, 2, and so on. Note that average ranks are assigned in case of ties. The average rank of one algorithm is obtained by averaging over all of data sets. Refer to Table VIII for the mean rank of these algorithms. The computational environment is Windows XP with Intel Core 2 Duo 1.66G CPU and 2-GB RAM. A MATLAB support vector machine toolbox [6] has been used to implement an SVM, in which SMO algorithm is implemented by C++ MEX files. This is the reason why SVM always runs faster than RVM and PCVM. The source code of RVM is obtained from Tipping's website.[12] PCVM is implemented in MATLAB.

IV. SOME THEORETICAL DISCUSSIONS ON PCVMS

According to the experimental results, PCVMs outperform RVMs and SVMs on most of the data sets. Section I presented some intuitive explanations for using truncated Gaussian prior in PCVMs. This section will discuss the reasons why PCVMs are better in our experiments using MAP analysis and margin analysis.

A. MAP Analysis

In Bayesian inference, the posterior of $\mathbf{w}$ and b is obtained by maximizing the product of likelihood $p(y|\mathbf{w}, b)$ and prior $p(\mathbf{w}|\alpha)p(b|\beta)$, where α is the parameter of the prior $\mathbf{w}$ and β is the parameter of the prior b. Since two kinds of likelihoods, Bernoulli likelihood and Gaussian likelihood, are often used in classification settings, we analyze these two cases, respectively.

1) Bernoulli Likelihood: Bernoulli likelihood is defined as follows:

$$p(y|\mathbf{w}, b) = \prod_{i=1}^{N} \Psi\left(f(\mathbf{x}_i; \mathbf{w})\right)^{t_i} \left[1 - \Psi\left(f(\mathbf{x}_i; \mathbf{w})\right)\right]^{1-t_i}$$

where $f(\mathbf{x}_i; \mathbf{w}) = \mathbf{\Phi}_\theta(\mathbf{x}_i)\mathbf{w} + b$, t_i is the target probability, $t_i \in \{0, 1\}$, and t_i is obtained by $t_i = (y_i + 1)/2$.

We make the common choice of a zero-mean Gaussian prior distribution over $\mathbf{w}$ and b

$$p(\mathbf{w}|\alpha)p(b|\beta) = \prod_{i=1}^{N} \left(\frac{\alpha_i}{2\pi}\right)^{1/2} \left(\frac{\beta}{2\pi}\right)^{1/2} \cdot \exp\left(-\frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} - \frac{1}{2}\beta b^2\right) \quad (16)$$

where $\mathbf{A}$ is a diagonal matrix and $\mathbf{A} = \text{diag}(\alpha_1, \cdots, \alpha_N)$, and α_i, β are inverse variance of the Gaussian distribution.

As the posterior $\mathbf{w}$ and b are proportional to the product of likelihood $p(y|\mathbf{w}, b)$ and prior $p(\mathbf{w}|\alpha)p(b|\beta)$, the MAP solution is equivalent to maximizing the following function:

$$\max_{\mathbf{w},b} \Upsilon_1 = \prod_{i=1}^{N} \Psi\left(f(\mathbf{x}_i)\right)^{t_i} \left[1 - \Psi\left(f(\mathbf{x}_i)\right)\right]^{1-t_i} \cdot \exp\left(-\frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} - \frac{1}{2}\beta b^2\right). \quad (17)$$

Taking the negative logarithm of (17), the maximum posterior is obtained as the solution to the following *minimization* problem:

$$\min_{\mathbf{w},b} \Upsilon_2(\mathbf{w}, b) = \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} + \frac{1}{2}\beta b^2 - \sum_{i=1}^{N} \left\{t_i \ln \Psi\left(f(\mathbf{x}_i)\right) + (1 - t_i) \ln\left[1 - \Psi\left(f(\mathbf{x}_i)\right)\right]\right\}.$$

The optimal solution of $\mathbf{w}$ can be obtained as follows:

$$\frac{\partial \Upsilon_2}{\partial \mathbf{w}} = 0 \longrightarrow \mathbf{w} = \sum_{i=1}^{N} \frac{\Psi'\left(f(\mathbf{x}_i)\right)\left(t_i - \Psi\left(f(\mathbf{x}_i)\right)\right)}{\Psi\left(f(\mathbf{x}_i)\right)\left(1 - \Psi\left(f(\mathbf{x}_i)\right)\right)} A^{-1}\mathbf{\Phi}_\theta^T(\mathbf{x_i}). \quad (18)$$

2) Gaussian Likelihood: The Gaussian likelihood is obtained as

$$p(y|\mathbf{w}, b) = \left(\frac{\delta}{2\pi}\right)^{N/2} \exp\left(-\frac{1}{2}\delta \sum_{i=1}^{N} \left(\Psi\left(f(\mathbf{x}_i; \mathbf{w})\right) - t_i\right)^2\right)$$

[11]Since the running time of SVM_{hard} and SVM_{PCVM} is similar to that of SVM_{soft}, we only record the running time of SVM_{soft}.

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

where δ is the inverse variance of $t_i - \Psi(f(\mathbf{x}_i)), i = 1, \cdots, N$. We take the same Gaussian prior (16) as in the previous case.

The maximization of posterior is equivalent to *minimizing* the following optimization problem:

$$\min_{\mathbf{w},b} \Upsilon_3 = \sum_{i=1}^{N} \left(\Psi\left(f(\mathbf{x}_i)\right) - t_i\right)^2 + \mathbf{w}^T\mathbf{C}\mathbf{w} + Tb^2$$

where $\mathbf{C} = \mathbf{A}/\delta$, $T = \beta/\delta$, and the optimization problem only depends on these ratios $\mathbf{C} = \mathbf{A}/\delta$ and $T = \beta/\delta$. The optimal $\mathbf{w}$ can be obtained as follows:

$$\frac{\partial \Upsilon_3}{\partial \mathbf{w}} = 0 \longrightarrow \mathbf{w} = \sum_{i=1}^{N} \left(t_i - \Psi\left(f(\mathbf{x}_i)\right)\right) \Psi'\left(f(\mathbf{x}_i)\right) C^{-1} \mathbf{\Phi}_\theta^T(\mathbf{x_i}). \tag{19}$$

All of the link functions, including sigmoid link or probit link, are monotonically increasing functions, and thus the slope is positive, meaning the function $\Psi'(f(\mathbf{x}_i)) > 0$. According to (18) and (19), $c_i = \alpha_i/\delta$, α_i, and $\Psi'(f(\mathbf{x}_i))$ are all nonnegative.

If we have a sparse model and a localized basis function $\mathbf{\Phi}$ (such as Gaussian used in this paper), then the expression for w_i will be dominated by the term $\phi_{i,\theta}(\mathbf{x}_i)$ and the sign of w_i will follow that of $(t_i - \Psi(f(\mathbf{x}_i))$. Since the bound of the link function $0 \leq \Psi(f(\mathbf{x}_i)) \leq 1$ and the t_i is mapped from y_i by the equation $t_i = (y_i + 1)/2$, w_i will have the same sign (or zero) as y_i.

B. Margin Analysis

The superiority of PCVMs' formulation can be analyzed by the concept of margin. Margin is first used by SVMs to enlarge the distance between the positive and negative classes. Then, Breiman [2] defined the margin for single points and used margin to analyze boosting algorithms. Other work on margin includes an explanation of Adaboost as boosting the margin [25] and construction of the soft-margin Adaboost [23].

In this paper, we follow the most common definition of margin [23], [25] for an input–output pair (x_i, y_i) by

$$m_i = y_i f(\mathbf{x}_i)$$

where $y_i = \{-1, +1\}$ and $f(\mathbf{x}_i) \in [-1, 1]$, $i = 1, \cdots, N$, and N denotes the number of training patterns. The margin at $\mathbf{x}_i$ is positive if the correct class label of the pattern is predicted. As the positivity of the margin value increases, the decision stability becomes larger. Moreover, as $f(\mathbf{x}_i) \in [-1, 1]$, $m_i \in [-1, 1]$. In the following, we analyze the Bernoulli likelihood and Gaussian likelihood, respectively.

1) Bernoulli Likelihood:

a) Gaussian Prior formulation: The optimal solution of $\mathbf{w}^{\text{Gauss}}$ is obtained by (18).

b) PCVM formulation: PCVMs incorporate a truncated Gaussian prior. Therefore, the maximum posterior is obtained as the solution to the following *minimization* problem:

$$\min_{\mathbf{w},b} \Upsilon_4(\mathbf{w}, b) = \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} + \frac{1}{2}\beta b^2 - \sum_{i=1}^{N} \left\{t_i \ln \Psi\left(f(\mathbf{x}_i)\right) + (1 - t_i) \ln\left[1 - \Psi\left(f(\mathbf{x}_i)\right)\right]\right\}$$

subject to $w_i y_i \geq 0, \qquad i = 1, \cdots, N.$

Therefore, we construct the Lagrange

$$\min_{\mathbf{w},b} L_1(\mathbf{w}, b) = \frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w} + \frac{1}{2}\beta b^2 - \sum_{i=1}^{N} \xi_i w_i y_i - \sum_{i=1}^{N} \left\{t_i \ln \Psi\left(f(\mathbf{x}_i)\right) + (1 - t_i) \ln\left[1 - \Psi\left(f(\mathbf{x}_i)\right)\right]\right\}$$

by introducing Lagrange multipliers $\xi_i \geq 0$ $(i = 1, \cdots N)$. The optimal weight $\mathbf{w}$ is obtained by solving the Lagrange problem

$$\frac{\partial L_1}{\partial \mathbf{w}} = 0 \longrightarrow$$

$$\mathbf{w}^{\text{pcvm}} = \sum_{i=1}^{N} \frac{\Psi'\left(f(\mathbf{x}_i)\right)\left(t_i - \Psi\left(f(\mathbf{x}_i)\right)\right)}{\Psi\left(f(\mathbf{x}_i)\right)\left(1 - \Psi\left(f(\mathbf{x}_i)\right)\right)} A^{-1}\mathbf{\Phi}_\theta^T(\mathbf{x_i}) + A^{-1}\boldsymbol{\xi}Y \tag{20}$$

where $\boldsymbol{\xi} = \text{diag}(\xi_1, \cdots, \xi_N)$ and $Y = (y_1, \cdots, y_N)^T$.

Based on the definition of margin, the margins for any point i with Gaussian priors and truncated priors are presented as follows:

$$m_i^{\text{Gauss}} = y_i \left(2\Psi\left(\mathbf{\Phi}_\theta(\mathbf{x}_i)\mathbf{w}^{\text{Gauss}} + b\right) - 1\right)$$
$$m_i^{\text{pcvm}} = y_i \left(2\Psi\left(\mathbf{\Phi}_\theta(\mathbf{x}_i)\mathbf{w}^{\text{pcvm}} + b\right) - 1\right)$$

where the transformation $2\Psi(\mathbf{\Phi}_\theta(\mathbf{x}_i)\mathbf{w} + b) - 1$ is to map the output $\Psi(\mathbf{\Phi}_\theta(\mathbf{x}_i)\mathbf{w} + b) \in [0, 1]$ to the desired range $[-1, 1]$

$$m_i^{\text{pcvm}} - m_i^{\text{Gauss}} = 2y_i \left\{\Psi\left(\mathbf{\Phi}_\theta(\mathbf{x}_i)\mathbf{w}^{\text{pcvm}} + b\right) - \Psi\left(\mathbf{\Phi}_\theta(\mathbf{x}_i)\mathbf{w}^{\text{Gauss}} + b\right)\right\}.$$

According to (18) and (20), as all the link functions are monotonically increasing function and the matrix $A^{-1}\boldsymbol{\xi} \geq 0$, the difference between the margins is decided by the term Y on the right-hand side of (20). $m_i^{\text{pcvm}} - m_i^{\text{Gauss}} \geq 0$ will be satisfied with a localized basis function $\mathbf{\Phi}$ (such as Gaussian function) in a sparse model.

2) Gaussian Likelihood: The maximum of the posterior is obtained as the solution to the following *minimization* problem in PCVMs:

$$\min_{\mathbf{w},b} \Upsilon_5(\mathbf{w}, b) = \sum_{i=1}^{N} \left(\Psi\left(f(\mathbf{x}_i)\right) - t_i\right)^2 + \mathbf{w}^T\mathbf{C}\mathbf{w} + Tb^2$$

subject to $w_i y_i \geq 0$.

Therefore, one constructs the Lagrange

$$\min_{\mathbf{w},b} L_2(\mathbf{w}, b) = \sum_{i=1}^{N} \left(\Psi\left(f(\mathbf{x}_i)\right) - t_i\right)^2 + \mathbf{w}^T\mathbf{C}\mathbf{w} + Tb^2 - \sum_{i=1}^{N} \xi_i w_i y_i$$

by introducing Lagrange multipliers $\xi_i \geq 0$ $(i = 1, \cdots N)$. The optimal weight vector $\mathbf{w}$ is obtained by solving the Lagrange problem

$$\frac{\partial L_2}{\partial \mathbf{w}} = 0 \longrightarrow$$

$$\mathbf{w}^{\text{pcvm}} = \sum_{i=1}^{N} \left(t_i - \Psi\left(f(\mathbf{x}_i)\right)\right) \Psi'\left(f(\mathbf{x}_i)\right) \mathbf{C}^{-1}\mathbf{\Phi}_\theta^T(\mathbf{x_i}) + \mathbf{C}^{-1}\boldsymbol{\xi}Y.$$

Following the same analysis as adopted in the previous section, PCVMs are better than RVMs in terms of margin with a localized basis function $\mathbf{\Phi}$ (such as Gaussian function used in this paper) in a sparse model.

C. Summary

This section analyzes the formulation of PCVMs using MAP analysis and margin analysis. Both analysis indicate that different truncated priors for different classes used in PCVMs are better than Gaussian priors in a sparse model with a localized basis function. This theoretical observation explains well the empirical success of PCVMs in this paper and strengthens the significance of this algorithm.

V. Conclusion

In this paper, a probabilistic algorithm, probabilistic classification vector machines (PCVMs), has been proposed for classification problems. The paper analyzes RVMs for classification problems and observes that adopting the same prior for different classes may lead to unstable solutions.

In order to tackle this problem, a signed and truncated Gaussian prior is adopted over every weight, where the sign of the prior is determined by the class label. Our algorithm benefits from the prior because it not only introduces the sparsity but also restricts the sign of every weight, which is suitable for classification problems. An efficient procedure for parameter optimization has been incorporated in the EM algorithm for PCVMs.

We have conducted a comprehensive study of PCVMs on four synthetic data sets and 13 benchmark problems under three performance metrics to explore the characteristics of PCVMs, SVMs, RVMs, and other algorithms. In order to compare these classifiers, several kinds of statistical tests have been done. The 5×2 cv F test [1] is used to compare paired classifiers on single data sets. To compare classifiers on multiple data sets, the Friedman test with the corresponding post-hoc test has been used to statistically compare these classifiers over multiple data sets.

Our results confirm that the PCVM performs very well on these data sets under all three metrics, especially under AUC. For the RVM, it appears that adopting the same prior from regression for classification problems leads to suboptimal results under ERR, AUC, and RMSE. The difference between the PCVM and the RVM shows that adopting truncated priors for different classes is beneficial.

This paper also discusses PCVMs using MAP analysis and margin analysis. Both analyses indicate that truncated priors in PCVMs are better than common Gaussian priors in a sparse model with a localized basis function. This theoretical finding explains well the empirical success of PCVMs and also strengthens the significance of this algorithm.

In general, we could conclude that the PCVM is a sparse learning algorithm that addresses the substantial drawbacks of SVMs without degrading the generalization performance. The PCVM provides probabilistic outputs to assess the uncertainty for the predictions and performs well on the skewed data sets, which are the reasons why the PCVM is so good under the AUC metric. The PCVM also incorporates an efficient parameter optimization procedure, not only saving the effort to do cross-validation grid search but also improving the performance. The interesting point here is that the PCVM-optimized parameter works for SVMs as well, providing an alternative to the usual parameter selection method for SVMs. The number of basis functions in PCVMs does not grow linearly with the number of training points, leading to simpler and easier to understand models.

The computational complexity of PCVMs is $O(M^3)$, where M is the number of *nonzero* basis functions and $M \leq N$. Because of the sparseness-inducing priors and fast converging EM algorithm, PCVMs prune the basis functions rapidly for most problems. The computation time of PCVMs is further reduced by their efficient parameter optimization procedure.

Future work for this study includes a more in-depth study of methods to tackle the local maxima problem in EM algorithm and reduction of computational complexity on large data sets.

Appendix

A. Further Details of Hierarchical Hyperpriors

To follow the Bayesian framework and encourage the model sparsity, hierarchical hyperpriors over α and β will be defined. In order to facilitate the comparison with the RVM, we use gamma distribution as the hyperprior. However, the hyperpriors are not restricted to gamma distribution. For example, the exponential distribution can also be employed as hyperpriors to introduce a Laplacian prior [12]

$$p(\alpha) = \prod_{i=1}^{N} \mathrm{Gamma}(\alpha_i | c, d)$$
$$p(\beta) = \mathrm{Gamma}(\beta | e, f)$$

where c, d, e, and f are parameters of the Gamma hyperprior and

$$\mathrm{Gamma}(\alpha_i | c, d) = \Gamma(c)^{-1} d^c \alpha_i^{c-1} e^{-d\alpha_i}$$

where $\Gamma(c) = \int_0^\infty t^{c-1} e^{-t} dt$ is the gamma function.

With these assumptions in place, the complete prior can be obtained by marginalizing with respect to each α_i and b

$$p(w_i|c,d) = \int_0^\infty p(w_i|\alpha_i) p(\alpha_i|c,d) d\alpha_i = \begin{cases} \dfrac{2d^c \Gamma\left(c+\frac{1}{2}\right)}{\sqrt{2\pi}\Gamma(c)} \left(\dfrac{w_i^2}{2}+d\right)^{-\left(c+\frac{1}{2}\right)}, & \text{if } y_i w_i \geq 0 \\ 0, & \text{if } y_i w_i < 0 \end{cases} \tag{21}$$

$$p(b|e,f) = \int_0^\infty p(b|\beta) p(\beta|e,f) d\beta = \frac{f^e \Gamma\left(e+\frac{1}{2}\right)}{\sqrt{2\pi}\Gamma(e)} \left(\frac{b^2}{2}+f\right)^{-\left(e+\frac{1}{2}\right)}. \tag{22}$$

According to (21) and (22), the hierarchical prior is equivalent to a truncated student-t prior over $\mathbf{w}$ and a student-t prior over b. This prior is sharply peaked at zero and more peaky than a Gaussian prior.

In most cases, the parameters c, d, e, and f will be set to zero. In this situation, a prior

$$p(w_i) = \begin{cases} \dfrac{2\Gamma\left(\frac{1}{2}\right)}{\sqrt{2\pi}\Gamma(0)} \left(\dfrac{w_i^2}{2}\right)^{-\frac{1}{2}}, & \text{if } y_i w_i \geq 0 \\ 0, & \text{if } y_i w_i < 0 \end{cases}$$

is obtained. The prior looks like the Laplacian prior and leads to a sparse model.

B. Details of Expectation Step

In the expectation step, we need to calculate the expectations of log-posterior (6) with respect to the latent variables. According to the definition, the expectation step can be obtained by the following formula:

$$\begin{aligned} &Q(\mathbf{w}, b|\mathbf{w}^{\text{old}}, b^{\text{old}}) \\ &= E_{\mathbf{H}_\theta,\alpha,\beta}\left[\log p(\mathbf{w}, b|\mathbf{y}, \mathbf{H}_\theta, \alpha, \beta)|y, \mathbf{w}^{\text{old}}, b^{\text{old}}\right] \\ &= 2\mathbf{w}^T\mathbf{\Phi}_\theta^T E[\mathbf{H}_\theta|y, \mathbf{w}^{\text{old}}, b^{\text{old}}] - \mathbf{w}^T\mathbf{\Phi}_\theta^T\mathbf{\Phi}_\theta\mathbf{w} - 2bI^T\mathbf{\Phi}_\theta\mathbf{w} \\ &\quad + 2bI^T E[\mathbf{H}_\theta|y, \mathbf{w}^{\text{old}}, b^{\text{old}}] - b^2 N \\ &\quad - \mathbf{w}^T E[\mathbf{A}|y, \mathbf{w}^{\text{old}}, b^{\text{old}}]\mathbf{w} - E[\beta|y, \mathbf{w}^{\text{old}}, b^{\text{old}}]b^2. \end{aligned}$$

The computation of $Q(\mathbf{w}, b|\mathbf{w}^{\text{old}}, b^{\text{old}})$ reduces to computing the expectations: $E[\mathbf{H}_\theta|y, \mathbf{w}^{\text{old}}, b^{\text{old}}]$, $E[\mathbf{A}|y, \mathbf{w}^{\text{old}}, b^{\text{old}}]$, and $E[\beta|y, \mathbf{w}^{\text{old}}, b^{\text{old}}]$

$$\begin{aligned} \bar{h}_{\theta,i} &= E\left[h_\theta(\mathbf{x}_i)|y_i, \mathbf{w}^{\text{old}}, b^{\text{old}}\right] \\ &= \int h_\theta(\mathbf{x}_i) \cdot p\left(h_\theta(\mathbf{x}_i)|y_i, \mathbf{w}^{\text{old}}, b^{\text{old}}\right) dh_\theta(\mathbf{x}_i) \\ &= \begin{cases} z_i\Psi(z_{\theta,i}) + N(z_{\theta,i}|0,1), & \text{if } y_i = +1 \\ z_i\Psi(-z_{\theta,i}) - N(z_{\theta,i}|0,1), & \text{if } y_i = -1 \end{cases} \end{aligned} \quad (23)$$

where $z_{\theta,i} = \mathbf{\Phi}_\theta(\mathbf{x}_i)\mathbf{w} + b$.

Note that the function of y_i in (23) is to restrict the integral bound: when $y_i = +1$, $p(h_\theta(\mathbf{x}_i)|y_i, \mathbf{w}^{\text{old}}, b^{\text{old}})$ is a left-truncated Gaussian from zero to infinity with mean $\mathbf{\Phi}_\theta(\mathbf{x}_i)\mathbf{w} + b$ and when $y_i = -1$, $p(h_\theta(\mathbf{x}_i)|y_i, \mathbf{w}^{\text{old}}, b^{\text{old}})$ is a right-truncated Gaussian from negative infinity to zero with mean $\mathbf{\Phi}_\theta(\mathbf{x}_i)\mathbf{w}+b$.

Since $\mathbf{A}$ is a diagonal matrix, $\mathbf{A} = \text{diag}(\alpha_1, \cdots, \alpha_N)$, the expectation $E[\mathbf{A}|y, \mathbf{w}^{\text{old}}, b^{\text{old}}]$ can be proceeded as a diagonal matrix $\bar{\mathbf{A}} = \text{diag}(E[\alpha_i|y_i, \mathbf{w}^{\text{old}}, b^{\text{old}}])$

$$\begin{aligned} \bar{\alpha}_i &= E[\alpha_i|y_i, \mathbf{w}^{\text{old}}, b^{\text{old}}] \\ &= \frac{\int_0^\infty \alpha_i \cdot p(w_i|\alpha_i)p(\alpha_i)d\alpha_i}{\int_0^\infty p(w_i|\alpha_i)p(\alpha_i)d\alpha_i} = \frac{c + 1/2}{w_i^2 + d} \end{aligned} \quad (24)$$

and

$$\begin{aligned} \bar{\beta} &= E[\beta|y_i, \mathbf{w}^{\text{old}}, b^{\text{old}}] \\ &= \frac{\int_0^\infty \beta \cdot p(b|\beta)p(\beta)d\beta}{\int_0^\infty p(b|\beta)p(\beta)d\beta} = \frac{e + 1/2}{b^2 + f}. \end{aligned} \quad (25)$$

Usually, we set $c = d = e = f = 0$.

Based on (23)–(25), the Q function is rewritten as follows:

$$\begin{aligned} Q(\mathbf{w}, b|\mathbf{w}^{\text{old}}, b^{\text{old}}) &= 2\mathbf{w}^T\mathbf{\Phi}_\theta^T\bar{\mathbf{H}}_\theta - \mathbf{w}^T\mathbf{\Phi}_\theta^T\mathbf{\Phi}_\theta\mathbf{w} + 2bI^T\bar{\mathbf{H}}_\theta \\ &\quad -2bI^T\mathbf{\Phi}_\theta\mathbf{w} - b^2N + \mathbf{w}^T\bar{A}\mathbf{w} - \bar{\beta}b^2 \end{aligned} \quad (26)$$

where $\bar{\mathbf{H}}_\theta$ is a vector of $\bar{h}_i$: $\bar{\mathbf{H}}_\theta = (\bar{h}_1, \bar{h}_2, \cdots, \bar{h}_N)^T$.

C. Further Details of Maximization Step

In the maximization step, we present the update rule for $\mathbf{w}$ and b

$$\mathbf{w}^{\text{new}} = \left(\mathbf{\Phi}_\theta^T\mathbf{\Phi}_\theta + \bar{\mathbf{A}}\right)^{-1}\left(\mathbf{\Phi}_\theta^T\bar{\mathbf{H}}_\theta - b\mathbf{\Phi}_\theta^T I\right) \quad (27)$$

$$b^{\text{new}} = \frac{I^T\bar{\mathbf{H}}_\theta - I^T\mathbf{\Phi}_\theta\mathbf{w}}{\bar{\beta} + N}. \quad (28)$$

From (24) and (25), the evaluation of $\bar{\alpha}_i$ and $\bar{\beta}$ needs to specify the parameters c, d, e, and f that are associated with hyperpriors. The model benefits from such hyperpriors by setting $c = d = e = f = 0$ since they are scale-invariant and such uniform hyperpriors have been shown to encourage model sparsity in [26]. This setting also facilitates comparison between the PCVM and the RVM since RVM uses the same hyperpriors and sets $c = d = e = f = 0$.

However, when setting these parameters to zero, the computation of $\bar{\alpha}_i(= 1/(2w_i^2))$ and $\bar{\beta}(= 1/(2b^2))$ is unstable when w_i's approach to zero. In our formulation, the diagonal matrix $\bar{\mathbf{A}}$ is updated in each M step. The elements of $\bar{\mathbf{A}}$ are inversely proportional to the square of the corresponding weights w_i: $\bar{\mathbf{A}} = \text{diag}[(\sqrt{2}w_1)^{-2}, \cdots, (\sqrt{2}w_N)^{-2}]$. Since some of the weights do eventually become small, it is not convenient to deal with $\bar{\mathbf{A}}$, because that would imply handling arbitrarily large numbers. We adopt a simple trick suggested in [12, Sec. 3.7, p. 1154] involving an auxiliary matrix $\mathbf{M} = \text{diag}[\sqrt{2}w_1, \cdots, \sqrt{2}w_N]$ and $(\mathbf{M}^{-1})^2 = \bar{\mathbf{A}}$. This transformation avoids the inversion of the elements of w_i when updating the weight parameters. The same modification is applied to (28) as well

$$\begin{aligned} \mathbf{w}^{\text{new}} &= \mathbf{M}\left(\mathbf{M}\mathbf{\Phi}_\theta^{\mathbf{T}}\mathbf{\Phi}_\theta\mathbf{M} + \mathbf{I}_N^{(\text{iden})}\right)^{-1}\mathbf{M}\left(\mathbf{\Phi}_\theta^{\mathbf{T}}\bar{\mathbf{H}}_\theta - b\mathbf{\Phi}_\theta^{\mathbf{T}}I\right) \\ b^{\text{new}} &= t(1 + tNt)^{-1}t(I^T\bar{\mathbf{H}}_\theta - I^T\mathbf{\Phi}_\theta\mathbf{w}) \end{aligned}$$

where $\mathbf{I}_N^{(iden)}$ is an N-dimensional identity matrix, the diagonal elements in the diagonal matrix $\mathbf{M}$ are

$$m_i = (\bar{\alpha}_i)^{-1/2} = \begin{cases} \sqrt{2}w_i, & \text{if } y_i w_i \geq 0 \\ 0, & \text{if } y_i w_i < 0 \end{cases}$$

and the scalar $t = \sqrt{2}|b|$. These modifications allow for a stable numerical computation in practice.

Moreover, as suggested by Tipping [26, App. B.1, p. 235], even though in theory the matrix $(\mathbf{\Phi}_\theta^T\mathbf{\Phi}_\theta + \bar{\mathbf{A}})$ is positive definite, it may become numerically singular when some diagonal elements in matrix $\bar{\mathbf{A}}$ tends towards very large values ($> \nu = 10^{12}$ in our experiments), i.e., some w_i tends to zero. In this experiments, we delete the appropriate column from $\mathbf{\Phi}$ to avoid ill-conditioning. A similar procedure of pruning has been adopted by Figueiredo [12] as well. In this context, ν^{-1} is the weight cutoff value for pruning kernels out of the model. Note that only kernels with very small associated weights will be pruned out of the model.

Since Cholesky decomposition is numerically stable [22], to enhance numerical stability, we follow Tipping [26] and use Cholesky decomposition instead of direct matrix inversion in our experiments.

REFERENCES

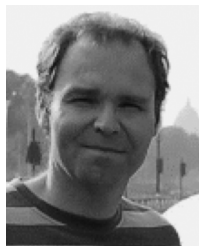
[1] E. Alpaydin, "Combined 5 × 2 cv f test for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 11, no. 8, pp. 1885–1892, 1999.

[2] L. Breiman, "Prediction games and arcing algorithms," *Neural Comput.*, vol. 11, no. 7, pp. 1493–1517, 1999.
[3] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Knowl. Disc. Data Mining*, vol. 2, no. 2, pp. 121–167, 1998.
[4] C. J. C. Burges and B. Schölkopf, "Improving the accuracy and speed of support vector learning machines," in *Advances in Neural Information Processing Systems 9 (NIPS)*. Cambridge, MA: MIT Press, 1997, pp. 375–381.
[5] R. Caruana and A. Niculescu-Mizil, "Data mining in metric space: An empirical analysis of supervised learning performance criteria," in *Proc. 10th Int. Conf. Knowl. Disc. Data Mining*, 2004, pp. 69–78.
[6] G. C. Cawley, "MATLAB Support Vector Machine Toolbox (v0.55 β)," Schl. Inf. Syst., Univ. East Anglia, Norwich, U.K., 2000 [Online]. Available: http://www.theoval.sys.uea.ac.uk/svm/toolbox
[7] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. B*, vol. 39, no. 1, pp. 1–38, 1977.
[8] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.
[9] T. G. Dietterich, "Approximate statistical test for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, 1998.
[10] O. J. Dunn, "Multiple comparisons among means," *J. Amer. Statist. Assoc.*, vol. 56, pp. 52–64, 1961.
[11] T. Fawcett, "An introduction to roc analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.
[12] M. A. T. Figueiredo, "Adaptive sparseness for supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 9, pp. 1150–1159, Sep. 2003.
[13] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Amer. Statist. Assoc.*, vol. 32, pp. 675–701, 1937.
[14] M. Friedman, "Comparison of alternative tests of significance for the problem of m rankings," *Ann. Math. Statist.*, vol. 11, pp. 86–92, 1940.
[15] R. L. Iman and J. M. Davenport, "Approximations of the critical region of the Friedman statistic," *Commun. Statist.*, pp. 571–595, 1980.
[16] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 169–184.
[17] D. J. C. MacKay, "Bayesian interpolation," *Neural Comput.*, vol. 4, no. 3, pp. 415–447, 1992.
[18] D. J. C. MacKay, "The evidence framework applied to classification networks," *Neural Comput.*, vol. 4, no. 3, pp. 720–736, 1992.
[19] P. McCullagh, *Generalized Linear Models*. London, U.K.: Chapman & Hall, 1989.
[20] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz, UCI Repository of Machine Learning Databases, Univ. California Irvine, Irvine, CA, 1998 [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
[21] J. Platt, "Probabilistic outputs for support vector machines and comparison to regularize likelihood methods," in *Advances in Large Margin Classifiers*, A. J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, Eds. Cambridge, MA: MIT Press, 2000, pp. 61–74.
[22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 2007.
[23] G. Rätsch, T. Onoda, and K. R. Müller, "Soft margins for adaboost," *Mach. Learn.*, vol. 42, no. 3, pp. 287–320, 2001.
[24] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.
[25] R. E. Schapire, Y. Freund, P. Barlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Ann. Statist.*, vol. 26, no. 5, pp. 1651–1686, 1998.
[26] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, 2001.
[27] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley-Interscience, 1998.

Huanhuan Chen (S'06–M'07) received the B.Sc. degree in electrical engineering 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 from University of Birmingham, Birmingham, U.K., in 2008.

He is a Research Fellow with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham. His research interests include statistical machine learning, data mining, and evolutionary computation.

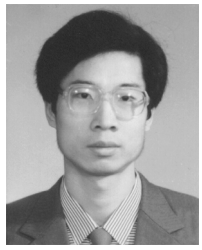
Dr. Chen is the recipient of the Value in People (VIP) award from The Wellcome Trust (2009), Dorothy Hodgkin Postgraduate Award (DHPA) from EPSRC (2004), and the Student Travel Grant for the 2006 Congress on Evolutionary Computation (CEC06).



Peter Tiňo received the M.Sc. degree from the Slovak University of Technology, Bratislava, Slovakia, in 1988 and the Ph.D. degree from the Slovak Academy of Sciences, Bratislava, Slovakia, in 1997, both in computer science.

He was a Fullbright Fellow at the NEC Research Institute, Princeton, NJ, from 1994 to 1995. He was a Postdoctoral Fellow at the Austrian Research Institute for AI, Vienna, Austria, from 1997 to 2000, and a Research Associate at the Aston University, U.K., from 2000 to 2003. Since 2003, he has been with the School of Computer Science, University of Birmingham, Birmingham, U.K., where he is currently a Senior Lecturer. His main research interests include probabilistic modeling and visualization of structured data, statistical pattern recognition, dynamical systems, evolutionary computation, and fractal analysis.

Dr. Tiňo is a recipient of the Fullbright Fellowship in 1994. He was awarded the Outstanding Paper of the Year for the IEEE TRANSACTIONS ON NEURAL NETWORKS with T. Lin, B. G. Horne, and C. L. Giles in 1998 for the work on recurrent neural networks. He won the 2002 Best Paper Award at the International Conference on Artificial Neural Networks with B. Hammer. He is on the editorial board of several journals.



Xin Yao (M'91–SM'96–F'03) received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, Anhui, China, in 1982, the M.Sc. degree from the North China Institute of Computing Technology, Beijing, China, in 1985, and the Ph.D. degree from USTC in 1990, all in computer science.

He was an Associate Lecturer and Lecturer from 1985 to 1990 at USTC, while working towards his Ph.D on simulated annealing and evolutionary algorithms. He took up a Postdoctoral Fellowship in the Computer Sciences Laboratory, Australian National University (ANU), Canberra, Australia, in 1990, and continued his work on simulated annealing and evolutionary algorithms. He joined the Knowledge-Based Systems Group, CSIRO (Commonwealth Scientific and Industrial Research Organisation) Division of Building, Construction and Engineering, Melbourne, Australia, in 1991, working primarily on an industrial project on automatic inspection of sewage pipes. He returned to Canberra in 1992 to take up a lectureship in the School of Computer Science, University College, University of New South Wales (UNSW), Australian Defence Force Academy (ADFA), where he was later promoted to a Senior Lecturer and Associate Professor. He moved to the University of Birmingham, U.K., as a Professor (Chair) of Computer Science in 1999. Currently, he is the Director of the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA) and a Changjiang (Visiting) Chair Professor (Cheung Kong Scholar) at the USTC. His major research interests include evolutionary artificial neural networks, automatic modularization of machine learning systems, evolutionary optimization, constraint handling techniques, computational time complexity of evolutionary algorithms, coevolution, iterated prisoner's dilemma, data mining, and real-world applications. He has more than 300 refereed publications.

Dr. Yao was the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION (2003–2008), an associate editor or editorial board member of 12 other journals, and the Editor of the World Scientific Book Series on *Advances in Natural Computation*. He has given more than 50 invited keynote and plenary speeches at conferences and workshops worldwide. He was awarded the President's Award for Outstanding Thesis by the Chinese Academy of Sciences for his doctoral work on simulated annealing and evolutionary algorithms in 1989. He won the 2001 IEEE Donald G. Fink Prize Paper Award for his work on evolutionary artificial neural networks.