Bayesian Trigonometric Support Vector Classifier

Wei Chu engp9354@nus.edu.sg

S. Sathiya Keerthi* mpessk@nus.edu.sg

Chong Jin Ong mpeongcj@nus.edu.sg

Control Division, Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore, 119260

Abstract

This paper describes Bayesian techniques for support vector classification. In particular, we propose a novel differentiable loss function called the trigonometric loss function which has the desirable characteristic of natural normalization in the likelihood function, and then follow standard Gaussian processes techniques to set up a Bayesian framework. In this framework, Bayesian inference is used to implement model adaptation, while keeping the merits of support vector classifier, such as sparseness and convex programming. This differs from standard Gaussian processes for classification. Moreover, we put forward class probability in making predictions. Experimental results on benchmark data sets indicate the usefulness of this approach.

1 Introduction

As a computationally powerful class of supervised learning networks, classical support vector classifier (SVC) (Vapnik, 1995) exploits the idea of mapping the input data into a high dimensional (often infinite) Hilbert space defined by a reproducing kernel

*All the correspondences should be addressed to S. Sathiya Keerthi.

(RKHS), where a linear classification is performed. The discriminant function is constructed by solving a regularized functional via convex quadratic programming. The advantages of classical SVC are: a global minimum solution; relatively fast training speed for large-scale learning tasks; and sparseness in solution representation. The choice of the regularization parameter and the other kernel parameters in the SVC model crucially affect the generalization performance. Model selection is usually based on the criterion of some simple and pertinent performance measures, such as cross validation (Wahba, 1990) or various generalization bounds derived from statistical learning theory (Vapnik, 1995). Typically, Bayesian methods are regarded as suitable tools to determine the values of these parameters. Moreover, Bayesian methods can also provide probabilistic class prediction that is more desirable than just deterministic classification.

There is some literature on Bayesian interpretations of classical SVC. Kwok (2000) built up MacKay's evidence framework (MacKay, 1992) using a weight-space interpretation. The unnormalized evidence may cause inaccuracy in Bayesian inference. Sollich (2002) pointed out that the normalization issue in Bayesian framework for classical SVC is critical and proposed an intricate Bayesian treatment with normalized evidence and error bar, where the evidence normalization depends on an unknown input distribution that limits its usefulness in practice.

In this paper, we introduce a novel loss function for SVC, called the trigonometric loss function, with the purpose of integrating Bayesian inference with SVC smoothly while preserving their individual merits. The trigonometric loss function is smooth and naturally normalized in likelihood evaluation. Further, it possesses the desirable property of sparseness in sample selection. We follow standard Gaussian processes for classification (Williams and Barber, 1998) to set up a Bayesian framework. Maximum a posteriori (MAP) estimate of the latent functions results in a convex programming problem. The popular sequential minimal optimization algorithm could be easily adapted to find the solution. Optimal parameters can then be inferred by Bayesian techniques.

The important advantages of our Bayesian treatment on SVC using the trigonometric loss function (BTSVC) over classical SVC are: (1) the capability to intrinsically and efficiently implement feature selection in the probabilistic framework; and (2) quite better generalization performance on sparse training sets. The probabilities

for class prediction that BTSVC provides are also rooted in the probabilistic framework. Compared with GPC, BTSVC possesses the sparseness property that reduces the computational burden and then helps us to tackle large data sets.

The paper is organized as follows: in section 2 we review the popular loss functions for binary classification, and then propose the trigonometric loss function; in section 3 we describe the Bayesian framework, formulate the MAP estimate on function values as a convex programming problem, and then evidence approximation can be applied to implement hyperparameter inference; in section 4 we discuss probabilistic class prediction; in section 5 we show the results of numerical experiments that verify the approach and we conclude in section 6.

2 Trigonometric Loss Function

In the probabilistic approach for binary classification, logistic function is widely used in likelihood evaluation (Williams and Barber, 1998), which is defined as

$$\mathcal{P}(y_x|f_x) = \frac{1}{1+\exp(-y_x \cdot f_x)} \tag{1}$$

where the input vector $x \in \mathbb{R}^d$, the class label $y_x \in \{+1, -1\}$ and f_x denotes the latent function (discriminant function) at x. $-\ln \mathcal{P}(y_x|f_x)$ is usually referred to as the loss function. The loss function associated with the shifted heaviside step function in SVC is also called the hard margin loss function, which is defined as

$$\ell_h(y_x \cdot f_x) = \begin{cases} 0 & if \;\; y_x \cdot f_x \geq +1; \\ +\infty & otherwise. \end{cases} \tag{2}$$

The hard margin loss function is suitable for noise-free data sets. For other general cases, a soft margin loss function is popularly used in SVC (Burges, 1998), which is defined as

$$\ell_\rho(y_x \cdot f_x) = \begin{cases} 0 & if \;\; y_x \cdot f_x \geq +1; \\ (1 - y_x \cdot f_x)^\rho & otherwise, \end{cases}$$

where ρ is a positive integer. The corresponding likelihood function in probabilistic framework can be written as

$$\mathcal{P}(y_x|f_x) = \frac{1}{\nu(f_x)} \cdot \exp(-\ell_\rho(y_x \cdot f_x)),$$

where $y_x \in \{-1, +1\}$ and the normalizer should be $\nu(f_x) = \exp(-\ell_\rho(+f_x))+\exp(-\ell_\rho(-f_x))$. Notice that the normalizer $\nu(f_x)$ is dependent on the latent function f_x. To compute the MAP estimate on function values f_x in Bayesian inference, the normalizer term $\nu(f_x)$ has to be taken into account that makes it inconvenient to find the solution. This flaw precludes the solution of SVC from being directly used as the MAP estimate (Sollich, 2002). Soft margin loss function is special in that it gives identical zero penalty to training samples that have satisfied the constraint $y_x \cdot f_x \geq +1$. These training samples are not involved in the Bayesian inference computations. This simplification of computational burden is usually referred to as the sparseness property. Logistic function does not enjoy this property since it contributes a positive penalty to all the training samples. On the other hand, logistic function is attractive because it is naturally normalized in likelihood evaluation, i.e., the normalizer is a constant, a property that allows Bayesian techniques to be used smoothly.

Based on these observations, we list desirable characteristics of a loss function for classification: it should be naturally normalized in likelihood evaluation; it should possess a flat zero region that results in sparseness property; it should be smooth and its first order derivative should be monotonic and continuous. Adhering to these requirements, we propose a novel loss function for binary classification, known as trigonometric loss function.[1] The trigonometric loss function is defined as

$$\ell_t(y_x \cdot f_x) = \begin{cases} +\infty & if \quad y_x \cdot f_x \in (-\infty, -1]; \\ 2\ln\sec(\frac{\pi}{4}(1 - y_x \cdot f_x)) & if \quad y_x \cdot f_x \in (-1, +1); \\ 0 & if \quad y_x \cdot f_x \in [+1, +\infty), \end{cases} \tag{3}$$

[1] It is possible to construct other loss functions with the desirable properties that the trigonometric loss function possesses, and even one with a continuous second order derivative.

The trigonometric likelihood function is therefore written as

$$\mathcal{P}_t(y_x|f_x) = \begin{cases} 0 & if \quad y_x \cdot f_x \in (-\infty, -1]; \\ \cos^2(\frac{\pi}{4}(1 - y_x \cdot f_x)) & if \quad y_x \cdot f_x \in (-1, +1); \\ 1 & if \quad y_x \cdot f_x \in [+1, +\infty). \end{cases} \tag{4}$$

The derivatives of the loss function are needed in the implementation of Bayesian methods. The first order derivative of (3) with respect to f_x can be derived as

$$\frac{\partial \ell_t(y_x \cdot f_x)}{\partial f_x} = \begin{cases} -y_x \frac{\pi}{2} \tan(\frac{\pi}{4}(1 - y_x \cdot f_x)) & if \quad y_x \cdot f_x \in (-1, +1); \\ 0 & if \quad y_x \cdot f_x \in [+1, +\infty), \end{cases} \tag{5}$$

and the second order derivative is

$$\frac{\partial^2 \ell_t(y_x \cdot f_x)}{\partial f_x^2} = \begin{cases} \frac{\pi^2}{8} \sec^2(\frac{\pi}{4}(1 - y_x \cdot f_x)) & if \quad y_x \cdot f_x \in (-1, +1); \\ 0 & if \quad y_x \cdot f_x \in [+1, +\infty). \end{cases} \tag{6}$$

From (4) and Figure 1, it is easy to see that the normalizer $\nu(f_x)$ is a constant for any f_x. From (3) and Figure 1, we find that the trigonometric loss function possesses a flat zero region that is same as the loss functions in classical SVC, but it requires that $y_x \cdot f_x > -1$ should always hold. One related issue for the trigonometric loss function is its sensitivity to outliers. We have conducted numerical experiments to understand this effect. It will be shown, in Section 5.1, that the general predictive ability using trigonometric loss function is not affected much by outliers, but only an increase in the number of support vectors is seen. Its generalization performance is found to be very close to the classical SVC method. The details are given in Section 5.

Remark 1 *The trigonometric loss function (3) can also be stated in a more general form as*

$$\ell_t(y_x \cdot f_x) = \begin{cases} +\infty & if \quad y_x \cdot f_x \in (-\infty, -\delta]; \\ 2 \ln \sec\left(\frac{\pi}{4}(1 - \frac{1}{\delta} \cdot y_x \cdot f_x)\right) & if \quad y_x \cdot f_x \in (-\delta, +\delta); \\ 0 & if \quad y_x \cdot f_x \in [+\delta, +\infty), \end{cases} \tag{7}$$

where $\delta > 0$. The optimal value of δ is determined by the noise level in training data.

3 Bayesian Inference

Introducing the trigonometric loss function into the regularized functional of classical SVC yields an optimization problem of minimizing the trigonometric SVC (TSVC) regularized functional in a RKHS

$$\min_{\boldsymbol{f}\in \mathrm{RKHS}} \mathcal{R}(\boldsymbol{f}) = \sum_{i=1}^{n} \ell_t(y_{x_i} \cdot f_{x_i}) + \lambda \|\boldsymbol{f}\|^2_{\mathrm{RKHS}}, \tag{8}$$

where the regularization parameter λ is positive and $\|\boldsymbol{f}\|^2_{\mathrm{RKHS}}$ is a norm in the RKHS. TSVC is derived along the way of classical SVC in the Appendix. Here we only focus on our initial motivation to integrate with Bayesian techniques. If we assume that the prior $\mathcal{P}(\boldsymbol{f}) \propto e^{-\lambda\|\boldsymbol{f}\|^2_{\mathrm{RKHS}}}$ and the likelihood $\mathcal{P}(\mathcal{D}|\boldsymbol{f}) \propto e^{-\sum_{i=1}^{n} \ell_t(y_{x_i} \cdot f_{x_i})}$, the minimizer of TSVC regularized functional (8) can be directly interpreted as maximum a posteriori (MAP) estimate of the function $\boldsymbol{f}$ in the RKHS (Evgeniou et al., 1999). Due to the duality between RKHS and stochastic processes (Wahba, 1990), the function $\boldsymbol{f}$ can also be explained as a family of random variables in a Gaussian process .

Recently, Gaussian processes have provided a promising non-parametric Bayesian approach to classification problems (Williams and Barber, 1998). The important advantage of Gaussian process models over other non-Bayesian models is the explicit probabilistic formulation. This not only gives the ability to infer model parameters in Bayesian framework but also provides probabilistic class prediction. We follow the standard Gaussian process classifier to describe a Bayesian framework, in which we impose a Gaussian process prior distribution on the latent functions and employ the trigonometric loss function in likelihood evaluation. Compared with standard Gaussian processes for classification, our approach uses the trigonometric loss function in place of the logistic loss function in likelihood evaluation; this results in a different convex programming problem for computing MAP estimate and leads to sparseness in computation. The TSVC classifier in Bayesian framework is referred to as Bayesian TSVC (BTSVC).

3.1 Bayesian Framework

The latent functions are usually assumed as the realizations of random variables indexed by the input vector x_i in a stationary zero-mean Gaussian process. The

Gaussian process can then be specified by giving the covariance matrix for any finite set of zero-mean random variables $\{f(x_i)|i = 1, 2, \ldots, n\}$. The covariance between the outputs corresponding to the inputs x_i and x_j can be defined as

$$Cov[f(x_i), f(x_j)] = \kappa_0 \exp\left(-\frac{1}{2}\kappa\|x_i - x_j\|^2\right) + \kappa_b, \tag{9}$$

where $\kappa_0 > 0$, $\kappa > 0$ and $\kappa_b > 0$. κ_0 denotes the average power of $f(x)$ that reflects the noise level. Note that the exponential term in (9) is exactly the Gaussian kernel in classical SVC,[2] while the second term corresponds to the variance of the offset in the latent functions. Thus the relationship between the covariance function and the kernel function is

$$Cov[f(x_i), f(x_j)] = \kappa_0 K(x_i, x_j) + \kappa_b, \tag{10}$$

where $K(x_i, x_j)$ denotes the Gaussian kernel function, i.e., $K(x_i, x_j) = \exp\left(-\frac{1}{2}\kappa\|x_i - x_j\|^2\right)$. Other kernel functions in classical SVC can also be used in the covariance function, such as polynomial kernels and spline kernels (Wahba, 1990). However, we only focus on Gaussian kernel in the present work.

We collect the parameters in the prior distribution $\{\kappa_0, \kappa, \kappa_b\}$, as θ, the hyperparameter vector. Thus, for a given hyperparameter vector θ, the prior probability of the random variables $\{f(x_i)\}$ is a multivariate Gaussian, which can be simply written as

$$\mathcal{P}(\boldsymbol{f}\,|\theta) = \frac{1}{Z_{\boldsymbol{f}}}\exp(-\frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f}), \tag{11}$$

where $\boldsymbol{f} = [f(x_1), f(x_2), \ldots, f(x_n)]^T$, $Z_{\boldsymbol{f}} = (2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}$, and Σ is the $n \times n$ covariance matrix whose ij-th element is $Cov[f(x_i), f(x_j)]$.

The likelihood with the trigonometric likelihood function (4) can be written as

$$\mathcal{P}(\mathcal{D}|\boldsymbol{f}, \theta) = \prod_{i=1}^{n} \mathcal{P}_t(y_{x_i}|f(x_i)). \tag{12}$$

[2]There is no need to multiply the term κ_0 in kernel function of classical SVC, due to the redundancy with the regularization parameter.

Based on Bayes' theorem, the posterior probability of $\boldsymbol{f}$ can then be written as

$$\mathcal{P}(\boldsymbol{f}\,|\mathcal{D},\theta) = \frac{1}{Z_S}\exp\left(-S(\boldsymbol{f})\right), \tag{13}$$

where $S(\boldsymbol{f}) = \frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f} + \sum_{i=1}^{n}\ell_t(y_{x_i}\cdot f(x_i))$, $\ell_t(\cdot)$ is defined as in (3) and $Z_S = \int\exp(-S(\boldsymbol{f}))\,d\boldsymbol{f}$. Since $\mathcal{P}(\boldsymbol{f}|\mathcal{D},\theta)\propto\exp(-S(\boldsymbol{f}))$, the MAP estimate on the values of $\boldsymbol{f}$ is therefore the minimizer of the following optimization problem

$$\min_{\boldsymbol{f}} S(\boldsymbol{f}) = \frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f} + \sum_{i=1}^{n}\ell_t(y_{x_i}\cdot f(x_i)). \tag{14}$$

This is a regularized functional. If $\boldsymbol{f}_{\mathrm{MP}}$ denotes an optimal solution of (14), then the derivative of $S(\boldsymbol{f})$ with respect to $\boldsymbol{f}$ should be zero at $\boldsymbol{f}_{\mathrm{MP}}$, i.e.,

$$\left.\frac{\partial S(\boldsymbol{f})}{\partial\boldsymbol{f}}\right|_{\boldsymbol{f}_{\mathrm{MP}}} = \Sigma^{-1}\cdot\boldsymbol{f} + \sum_{i=1}^{n}\left.\frac{\partial\ell_t(y_{x_i}\cdot f(x_i))}{\partial\boldsymbol{f}}\right|_{\boldsymbol{f}_{\mathrm{MP}}} = 0.$$

Let us now define the following set of unknowns: $\upsilon_i = -\left.\frac{\partial\ell_t(y_{x_i}\cdot f(x_i))}{\partial f(x_i)}\right|_{f_{\mathrm{MP}}(x_i)}$ where the derivative is as given in (5) and $\boldsymbol{\upsilon}$ as the column vector containing $\{\upsilon_i\}$. Then $\boldsymbol{f}_{\mathrm{MP}}$ can be written as:

$$\boldsymbol{f}_{\mathrm{MP}} = \Sigma\cdot\boldsymbol{\upsilon}. \tag{15}$$

Using (10), we can decompose the solution (15) into the form

$$f_{\mathrm{MP}}(x) = \sum_{i=1}^{n}\upsilon_i\cdot\kappa_0\cdot K(x,x_i) + \kappa_b\sum_{i=1}^{n}\upsilon_i, \tag{16}$$

to show the significance of the hyperparameters.[3] The hyperparameter κ_0 determines the average power of the patterns. The contribution of each pattern to the optimal discriminant function depends on its υ_i in (16). In the case of high noise level,

[3]Let us consider the covariance function $K(x_i,x_j)+\kappa_b$ and the general trigonometric loss function (7) in the regularized functional (14). Comparing the consequent solution with that in (16), we can notice that there is an equivalence between κ_0 in (10) and the parameter $1/\delta$ in (7).

a smaller value κ_0 can reduce the deleterious effect from some particular outliers. In the regularized functional (14), κ_0 in covariance function plays the role as the regularization parameter. κ_b is only involved in the bias term of the discriminant function (16).[4]

3.2 Convex Programming

In this subsection, we formulate the optimization problem (14) as a convex programming problem, and then adapt the popular sequential minimal optimization (SMO) algorithm (Platt, 1999; Keerthi et al., 2001) for the solution. As usual, slack variables ξ_i are introduced: $\xi_i \geq 1 - y_{x_i} \cdot f(x_i)$, $\forall i$. The optimization problem (14) can then be restated as the following equivalent optimization problem, which we refer to as the *primal* problem:

$$\min_{\boldsymbol{f},\boldsymbol{\xi}} \frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f} + 2\sum_{i=1}^{n} \ln \sec\left(\frac{\pi}{4}\xi_i\right) \tag{17}$$

subject to $y_{x_i} \cdot f(x_i) \geq 1 - \xi_i$ and $0 \leq \xi_i < 2$, $\forall i$. Standard Lagrangian techniques (Fletcher, 1987) are used to derive the *dual* problem. The strict inequality $\xi_i < 2$ is assumed to hold and omitted. As we will see below, this condition will be implicitly satisfied in the solution. Let $\alpha_i \geq 0$ and $\gamma_i \geq 0$ be the corresponding Lagrange multipliers for other inequalities in the *primal* problem (17). The Lagrangian for the *primal* problem (17) is:

$$L(\boldsymbol{f},\boldsymbol{\xi}) = \frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f} + 2\sum_{i=1}^{n} \ln \sec\left(\frac{\pi}{4}\xi_i\right) - \sum_{i=1}^{n}\gamma_i\cdot\xi_i - \sum_{i=1}^{n}\alpha_i(y_{x_i}\cdot f(x_i) - 1 + \xi_i). \tag{18}$$

The KKT conditions for the *primal* problem (17) are

$$f(x_i) = \sum_{j=1}^{n} y_{x_j}\alpha_j Cov[f(x_i), f(x_j)], \qquad \forall i; \tag{19}$$

$$\frac{\pi}{2}\tan\left(\frac{\pi}{4}\xi_i\right) = \alpha_i + \gamma_i, \qquad \forall i. \tag{20}$$

[4] κ_b might be trivial if the sum $\sum_{i=1}^{n} \upsilon_i$ is very small.

We can write (20) as

$$\xi_i = \frac{4}{\pi} \arctan\left(\frac{2}{\pi}(\alpha_i + \gamma_i)\right) \tag{21}$$

Given this, we note that the condition $\xi_i < 2$ is automatically satisfied. If we collect all the terms involving ξ_i in the Lagrangian (18), we get

$$T_i = 2 \ln \sec\left(\frac{\pi}{4}\xi_i\right) - (\alpha_i + \gamma_i)\xi_i.$$

Using (21) we can rewrite T_i as

$$T_i = \ln\left(1 + \left(\frac{2}{\pi}(\alpha_i + \gamma_i)\right)^2\right) - \frac{4}{\pi}(\alpha_i + \gamma_i) \arctan\left(\frac{2}{\pi}(\alpha_i + \gamma_i)\right). \tag{22}$$

Thus, the *dual* problem becomes a maximization problem involving only the dual variables, α_i and γ_i:

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\gamma}} \mathcal{R}(\boldsymbol{\alpha},\boldsymbol{\gamma}) = &-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(y_{x_i}\alpha_i)(y_{x_j}\alpha_j)Cov\left[f(x_i), f(x_j)\right] + \sum_{i=1}^{n}\alpha_i \\ &+ \sum_{i=1}^{n}\left[\ln\left(1 + \left(\frac{2}{\pi}(\alpha_i + \gamma_i)\right)^2\right) - \frac{4}{\pi}(\alpha_i + \gamma_i)\arctan\left(\frac{2}{\pi}(\alpha_i + \gamma_i)\right)\right] \end{aligned} \tag{23}$$

subject to

$$\alpha_i \geq 0 \text{ and } \gamma_i \geq 0, \forall i. \tag{24}$$

It is noted that $\mathcal{R}(\boldsymbol{\alpha},\boldsymbol{\gamma}) \leq \mathcal{R}(\boldsymbol{\alpha}, 0)$ for any $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ satisfying (24). Hence the maximization of (23) over $(\boldsymbol{\alpha},\boldsymbol{\gamma})$ satisfying (24) can be found by maximizing $\mathcal{R}(\boldsymbol{\alpha}, 0)$ over $\alpha_i \geq 0, \forall i$. Therefore, the *dual* problem can be finally simplified as

$$\begin{aligned} \min_{\boldsymbol{\alpha}} &\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(y_{x_i}\alpha_i)(y_{x_j}\alpha_j)Cov\left[f(x_i), f(x_j)\right] - \sum_{i=1}^{n}\alpha_i \\ &+ \sum_{i=1}^{n}\left[\frac{4}{\pi}\alpha_i \arctan\left(\frac{2\alpha_i}{\pi}\right) - \ln\left(1 + \left(\frac{2\alpha_i}{\pi}\right)^2\right)\right] \end{aligned} \tag{25}$$

subject to $\alpha_i \geq 0, \forall i$.

The *dual* problem (25) is a convex programming problem. In the following, we study the optimality conditions for the *dual* problem and adapt the popular SMO algorithm for the solution. Let $\eta_i \geq 0\ \forall i$ be the Lagrange multipliers corresponding to the inequalities in the *dual* problem (25). The KKT condition for the *dual* problem

(25) requires

$$F_i + y_{x_i}\eta_i = 0, \qquad \forall i, \tag{26}$$

where $F_i = -\sum_{j=1}^{n} y_{x_j}\alpha_j Cov\left[f(x_i), f(x_j)\right] + y_{x_i} - \frac{4}{\pi}\arctan\left(\frac{2y_{x_i}\alpha_i}{\pi}\right)$. The constraints (26) can be simplified by considering three cases for each i:

$$\begin{array}{lll} Case\ \ 1: & \alpha_i \neq 0 & F_i = 0; \\ Case\ \ 2: & \alpha_i = 0 \text{ and } y_{x_i} = -1 & F_i \geq 0; \\ Case\ \ 3: & \alpha_i = 0 \text{ and } y_{x_i} = +1 & F_i \leq 0. \end{array}$$

Any one pair could be classified into one of the three sets, which are defined as: $I_1 = \{i : \alpha_i \neq 0\}$, $I_2 = \{i : \alpha_i = 0 \text{ and } y_{x_i} = -1\}$, and $I_3 = \{i : \alpha_i = 0 \text{ and } y_{x_i} = +1\}$. Let us define $\beta^{up} = \min\{F_i : i \in I^{up}\}$ and $\beta^{low} = \max\{F_i : i \in I^{low}\}$, where $I^{up} = I_1 \cup I_2$ and $I^{low} = I_1 \cup I_3$. Optimality holds if $\beta^{up} \geq 0$ and $\beta^{low} \leq 0$. Thus, an approximate stopping condition is

$$\beta^{up} \geq -\tau \quad \text{and} \quad \beta^{low} \leq \tau \tag{27}$$

where τ is a positive tolerance parameter, say 10^{-3}. If (27) holds, we have reached a τ-optimal solution, and then the MAP estimate on the values of the random variables $\boldsymbol{f}$ can be determined from (19). We write (19) in column vector form as

$$\boldsymbol{f}_{\mathrm{MP}} = \Sigma \cdot \boldsymbol{v} \tag{28}$$

where $\boldsymbol{v} = [y_{x_1}\alpha_1, y_{x_2}\alpha_2, \ldots, y_{x_n}\alpha_n]^T$, that is consistent with the form (15). The training samples (x_i, y_{x_i}) associated with non-zero Lagrange multiplier α_i are called *support vectors* (SVs). The other samples associated with zero α_i do not involve in the solution representation and the following Bayesian computation. This property is usually referred to as sparseness, and it reduces the computational cost significantly.

The popular SMO algorithm for classical SVC (Platt, 1999; Keerthi et al., 2001) can be easily adapted to solve the optimization problem. The basic idea is to update the pair of Lagrange multipliers associated with β^{up} and β^{low} towards the minimum iteratively till the stopping condition (27) is satisfied. The difference is that the sub-optimization problem cannot be analytically solved. In the sub-optimization problem, we use multiple Newton-Raphson steps to update the two Lagrangian multipliers till convergence, then we return to SMO to update the variables β^{up} and β^{low}, and

check the stopping condition (27). The details of the Newton-Raphson steps are as follows. If α_i and α_j are being optimized in the sub-optimization problem, the Newton-Raphson formula for updating is

$$\begin{bmatrix} \alpha_i^{new} \\ \alpha_j^{new} \end{bmatrix} = \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} - \begin{bmatrix} H_{ii} & H_{ij} \\ H_{ij} & H_{jj} \end{bmatrix}^{-1} \cdot \begin{bmatrix} -y_{x_i} \cdot F_i \\ -y_{x_j} \cdot F_j \end{bmatrix} \tag{29}$$

where $H_{ii} = Cov[f(x_i), f(x_i)] + \frac{8}{\pi^2 + 4 \cdot \alpha_i^2}$, $H_{ij} = y_{x_i} \cdot y_{x_j} \cdot Cov[f(x_i), f(x_j)]$ and F_i is as defined below (26). We use this formula (29) to update α_i and α_j iteratively till the change in the α variables is less than 10^{-6} or the number of iterations is greater than 100. In numerical experiments, we found that the adapted algorithm can efficiently find the solution at nearly the same computational cost as that required by the quadratic programming in classical SVC. Of course, other methods for solving convex programming problems, such as dual subgradient schemes (Larsson et al., 1999) or interior point methods (Vanderbei, 2001), can also be used for the solution.

3.3 Hyperparameter Inference

The optimal values of hyperparameters θ can be inferred by maximizing the posterior probability $\mathcal{P}(\theta|\mathcal{D})$, using $\mathcal{P}(\theta|\mathcal{D}) = \mathcal{P}(\mathcal{D}|\theta)\mathcal{P}(\theta)/\mathcal{P}(\mathcal{D})$. A prior distribution on the hyperparameters $\mathcal{P}(\theta)$ is required here. As we typically have little idea about the suitable values of θ before training data are available, we assume a flat distribution for $\mathcal{P}(\theta)$, i.e., $\mathcal{P}(\theta)$ is greatly insensitive to the values of θ. Therefore, $\mathcal{P}(\mathcal{D}|\theta)$ (which is known as the evidence of θ) can be used to assign a preference to alternative values of the hyperparameters θ (MacKay, 1992). The evidence can be calculated by an explicit formula after using a Laplacian approximation at $\boldsymbol{f}_{\mathrm{MP}}$, and then hyperparameter inference may be done by gradient-based optimization methods.

The evidence is given by an integral over all $\boldsymbol{f}$: $\mathcal{P}(\mathcal{D}|\theta) = \int \mathcal{P}(\mathcal{D}|\theta, \boldsymbol{f})\mathcal{P}(\boldsymbol{f}|\theta)\, d\boldsymbol{f}$. Using the definitions in (11) and (12), the evidence can also be written as

$$\mathcal{P}(\mathcal{D}|\theta) = \frac{1}{Z_{\boldsymbol{f}}} \int \exp(-S(\boldsymbol{f}))\, d\boldsymbol{f}. \tag{30}$$

The marginalization can be done analytically by considering the Taylor expansion of $S(\boldsymbol{f})$ around its minimum $S(\boldsymbol{f}_{\mathrm{MP}})$, and retaining terms up to the second order. Since the first order derivative with respect to $\boldsymbol{f}$ at the most probable point $\boldsymbol{f}_{\mathrm{MP}}$ is zero,

$S(\boldsymbol{f})$ can be written as

$$S(\boldsymbol{f}) \approx S(\boldsymbol{f}_{\mathrm{MP}}) + \frac{1}{2}(\boldsymbol{f} - \boldsymbol{f}_{\mathrm{MP}})^T \left.\frac{\partial^2 S(\boldsymbol{f})}{\partial \boldsymbol{f} \partial \boldsymbol{f}^T}\right|_{\boldsymbol{f}=\boldsymbol{f}_{\mathrm{MP}}} (\boldsymbol{f} - \boldsymbol{f}_{\mathrm{MP}}), \tag{31}$$

where $\frac{\partial^2 S(\boldsymbol{f})}{\partial \boldsymbol{f} \partial \boldsymbol{f}^T} = \Sigma^{-1} + \Lambda$, and Λ is a diagonal matrix coming from the second order derivative of trigonometric loss function (6). Introducing (31) into (30) yields

$$\mathcal{P}(\mathcal{D}|\theta) = \exp(-S(\boldsymbol{f}_{\mathrm{MP}})) \cdot |\mathbf{I} + \Sigma \cdot \Lambda|^{-\frac{1}{2}},$$

where $\mathbf{I}$ is the $n \times n$ identity matrix. Notice that only a sub-matrix of Σ plays a role in the determinant $|\mathbf{I}+\Sigma\cdot\Lambda|$ due to the sparseness of the diagonal matrix Λ in which only the entries associated with SVs are non-zero. We denote their sub-matrices as Σ_{M} and Λ_{M} respectively by keeping their non-zero entries. The MAP estimate of $\boldsymbol{f}$ (28) on support vectors can also be simplified as $\boldsymbol{f}_{\mathrm{MP}} = \Sigma_{\mathrm{M}} \cdot \boldsymbol{\upsilon}_{\mathrm{M}}$, where $\boldsymbol{\upsilon}_{\mathrm{M}}$ denotes the sub-vector of $\boldsymbol{\upsilon}$ by keeping entries associated with SVs. Because of these sparseness properties, the negative log of the evidence can then be simplified as in the following remark.

Remark 2 *The negative logarithm of the evidence, which is the probability of data given hyperparameters $\mathcal{P}(\mathcal{D}|\theta)$, can be written as*

$$-\ln \mathcal{P}(\mathcal{D}|\theta) = \frac{1}{2}\boldsymbol{\upsilon}_{\mathrm{M}}^T \cdot \Sigma_{\mathrm{M}} \cdot \boldsymbol{\upsilon}_{\mathrm{M}} + 2 \sum_{m \in \mathrm{SVs}} \ln \sec\left(\frac{\pi}{4}\xi_m\right) + \frac{1}{2}\ln|\mathbf{I} + \Sigma_{\mathrm{M}} \cdot \Lambda_{\mathrm{M}}|, \tag{32}$$

where $\mathbf{I}$ is the identity matrix with the size of SVs, "$m \in SVs$" denotes that m is varied over the index set of SVs and $\xi_m = 1 - y_{x_m} \cdot f_{\mathrm{MP}}(x_m)$, $\forall m$.

The evidence evaluation is a convenient yardstick for model selection. Note that the evidence depends on the set of SVs. This set will change as the hyperparameters are varied. The evidence is a smooth function of the hyperparameters within the regions of hyperparameter space where the set of SVs remains unchanged.[5] We assume that

[5]It is possible that $-\ln \mathcal{P}(\mathcal{D}|\theta)$ has jumps at the points where the set of SVs changes discontinuously. This may lead to situations where gradient-based optimization algorithms perform poorly, since the local gradient could have a completely different direction from the "overall slope" of the evidence. However, in detailed

the set of SVs remains the same near the minimum of the evidence. The minimizer of $-\ln \mathcal{P}(\mathcal{D}|\theta)$ could then be inferred by some gradient-based optimization methods. We usually collect $\{\ln \kappa_0, \ln \kappa, \ln \kappa_b\}$ as the set of variables to tune, and the derivatives of (32) with respect to these variables are required. We give an expression for the derivatives of $-\ln \mathcal{P}(\mathcal{D}|\theta)$ with respect to these variables in the following remark.

Remark 3 *The derivatives of $-\ln \mathcal{P}(\mathcal{D}|\theta)$ with respect to the variables can be generally given as*

$$\begin{aligned}\frac{\partial - \ln \mathcal{P}(\mathcal{D}|\theta)}{\partial \ln \theta} = & \frac{\theta}{2} tr\left((\Lambda_{\mathrm{M}}^{-1} + \Sigma_{\mathrm{M}})^{-1} \frac{\partial \Sigma_{\mathrm{M}}}{\partial \theta}\right) - \frac{\theta}{2} \boldsymbol{v}_{\mathrm{M}}^T \frac{\partial \Sigma_{\mathrm{M}}}{\partial \theta} \boldsymbol{v}_{\mathrm{M}} \\ & -\frac{\theta}{2} \sum_{m \in \mathrm{SVs}} \boldsymbol{v}_{\mathrm{M}}^m \big((\Lambda_{\mathrm{M}}^{-1} + \Sigma_{\mathrm{M}})^{-1} \cdot \Sigma_{\mathrm{M}}\big)_{mm} \left(\Lambda_{\mathrm{M}}^{-1} (\Lambda_{\mathrm{M}}^{-1} + \Sigma_{\mathrm{M}})^{-1} \frac{\partial \Sigma_{\mathrm{M}}}{\partial \theta} \boldsymbol{v}_{\mathrm{M}}\right)^m \end{aligned} \quad (33)$$

where $\theta \in \{\kappa_0, \kappa, \kappa_b\}$, the subscript mm denotes the mm-th entry of a matrix, the superscript m denotes the m-th entry of a vector and, "$m \in$ SVs" denotes that m is varied over the index set of SVs.

In standard Gaussian processes for classification (Williams and Barber, 1998), the inversion of the full matrix Σ has to be computed in an iterative mode. This is a heavy burden for large-scale learning tasks. In our approach, only the inversion of the sub-matrix Σ_{M}, corresponding to the SVs, is required in the gradient evaluation (33). This sparseness in gradient evaluation makes it possible for our approach to tackle reasonably large data sets with thousands of samples, as the SVs usually form a small subset of the training samples.

3.4 Feature Selection

Automatic relevance determination (ARD) was proposed by MacKay (1994) and Neal (1996) as a hierarchical prior over the weights in neural networks. The weights connected to an irrelevant input can be automatically punished with a tighter prior in numerical experiments we did not experience any difficulties in finding the optimal solution using gradient-based algorithms on $-\ln \mathcal{P}(\mathcal{D}|\theta)$.

model adaptation, which reduces the influence of such a weight towards zero effectively. ARD can be directly embedded into the covariance function (9) as follows

$$Cov[f(x_i), f(x_j)] = \kappa_0 \exp\left(-\frac{1}{2}\sum_{\iota=1}^{d} \kappa_\iota (x_i^\iota - x_j^\iota)^2\right) + \kappa_b, \tag{34}$$

where x^ι denotes the ι-th entry of the input vector x, and κ_ι is the ARD parameter that determines the relevance of the ι-th input dimension to the target. The derivatives of $-\ln \mathcal{P}(\mathcal{D}|\theta)$ (32) with respect to the variables $\{\ln \kappa_\iota\}$ can be evaluated like we did in Remark 3.

Note that the training process with large number of hyperparameters can get stuck at local minima since we use gradient-based optimization methods. In general, we can perform the optimizations of the evidence several times starting from different initial states, and choose the one with the highest evidence as optimal choice while discarding other candidates. We can also organize these candidates together as an expert committee to represent the predictive distribution that can reduce the uncertainty with respect to the hyperparameters.

4 Probabilistic Class Prediction

In this section, we present details associated with the probabilistic class prediction on test patterns (MacKay, 1992; Bishop, 1995). This ability to provide the class probability is one of the important advantages of the probabilistic approach over the usual deterministic approach.

Let us take a test case x for which the class label y_x is unknown. The random variable $f(x)$ and the vector $\boldsymbol{f}$ containing the n zero-mean random variables $\{f(x_i)\}_{i=1}^n$ have the prior joint multivariate Gaussian distribution,

$$\begin{bmatrix} \boldsymbol{f} \\ f(x) \end{bmatrix} \sim N\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma & \boldsymbol{k} \\ \boldsymbol{k}^T & Cov[f(x), f(x)] \end{pmatrix}\right]$$

where $\boldsymbol{k} = [\,Cov[f(x), f(x_1)], \ldots, Cov[f(x), f(x_n)]\,]^T$. The conditional distribution of

$f(x)$ given $\boldsymbol{f}$ is a Gaussian:

$$\mathcal{P}(f(x)|\boldsymbol{f},\mathcal{D},\theta) \propto \exp\left(-\frac{1}{2}\frac{(f(x)-\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{k})^2}{Cov[f(x),f(x)]-\boldsymbol{k}^T\Sigma^{-1}\boldsymbol{k}}\right). \tag{35}$$

To remove the uncertainty in $\boldsymbol{f}$, we compute $\mathcal{P}(f(x)|\mathcal{D},\theta)$ by an integral over $\boldsymbol{f}$-space, which can be written as

$$\mathcal{P}(f(x)|\mathcal{D},\theta) = \int \mathcal{P}(f(x)|\boldsymbol{f},\mathcal{D},\theta)\mathcal{P}(\boldsymbol{f}\,|\mathcal{D},\theta)\,d\boldsymbol{f}, \tag{36}$$

where $\mathcal{P}(\boldsymbol{f}|\mathcal{D},\theta)$ is given as in (13). We make a Laplacian approximation on $S(\boldsymbol{f})$ at $\boldsymbol{f}_{\rm MP}$ as given by (31), and replace $\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{k}$ by its linear expansion around $\boldsymbol{f}_{\rm MP}$, i.e.,

$$\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{k} = \boldsymbol{f}_{\rm MP}^T\Sigma^{-1}\boldsymbol{k} + \boldsymbol{k}^T\Sigma^{-1}(\boldsymbol{f}-\boldsymbol{f}_{\rm MP}). \tag{37}$$

By computing the integral over $\boldsymbol{f}$ (36) with the approximation (31) and the linear expansion (37), the distribution of $\mathcal{P}(f(x)|\mathcal{D},\theta)$ can be evaluated as the Gaussian distribution

$$\mathcal{P}(f(x)|\mathcal{D},\theta) \sim \mathcal{N}(\mu_t,\sigma_t^2) = \frac{1}{\sqrt{2\pi}\sigma_t}\exp\left(-\frac{(f(x)-\mu_t)^2}{2\sigma_t^2}\right). \tag{38}$$

where the mean is $\mu_t = \boldsymbol{v}_{\rm M}^T\boldsymbol{k}_{\rm M}$, the variance is $\sigma_t^2 = Cov[f(x),f(x)] - \boldsymbol{k}_{\rm M}^T(\Lambda_{\rm M}^{-1} + \Sigma_{\rm M})^{-1}\boldsymbol{k}_{\rm M}$,[6] and $\boldsymbol{k}_{\rm M}$ is the sub-vector of $\boldsymbol{k}$ by keeping the entries associated with SVs. The standard deviation σ_t of the predictive distribution on x is also known as the error bar on the mean value μ_t. The second term in the σ_t^2 evaluation is a measure on the geometric distance between the test case x and the set of SVs in feature space. In other words, the test case x tends to get a broad predictive distribution if it lies far away from the SVs in feature space, and vice versa.

Now we make probabilistic class prediction. Given the hyperparameters θ, the probability of the binary class label y_x for the testing case x can be evaluated as:

$$\mathcal{P}(y_x|\mathcal{D},\theta) = \int \mathcal{P}(y_x|f(x),\mathcal{D},\theta)\mathcal{P}(f(x)|\mathcal{D},\theta)\,df(x)\,,$$

[6]The matrix inverse is already at hand after Bayesian inference with evidence gradient evaluations.

where $\mathcal{P}(y_x|f(x),\mathcal{D},\theta)$ is evaluated by trigonometric likelihood function (4) and $\mathcal{P}(f(x)|\mathcal{D},\theta)$ is given by (38). The one dimensional integral can be easily computed as:

$$\begin{aligned}\mathcal{P}(y_x|\mathcal{D},\theta) &= \frac{1}{2}\mathrm{erfc}\left(\frac{1-y_x\mu_t}{\sqrt{2}\sigma_t}\right) \\ &+ \int_{-1}^{+1} \cos^2\left(\frac{\pi}{4}(1-y_x f(x))\right)\mathcal{N}(\mu_t,\sigma_t^2)\,df(x)\,,\end{aligned} \tag{39}$$

where $\mathrm{erfc}(\nu) = \frac{2}{\sqrt{\pi}}\int_{\nu}^{+\infty}\exp(-z^2)\,dz$. The definite integral from -1 to $+1$ can be calculated by Romberg integration which may yield accurate results using much few function evaluations. Note that $\mathcal{P}(y_x=+1|\mathcal{D},\theta)$ is a monotonically increasing function of μ_t and $\mathcal{P}(y_x|\mathcal{D},\theta) = 0.5$ only when the predictive mean $\mu_t = 0$. However $\mathcal{P}(y_x|\mathcal{D},\theta)$ also depends on the predictive variance σ_t^2 for any $\mu_t \neq 0$. Specifically, $\mathcal{P}(y_x=+1|\mathcal{D},\theta)$ is a monotonically decreasing function of the predictive variance σ_t^2 when $\mu_t > 0$, but it is a monotonically increasing function of σ_t^2 when $\mu_t < 0$.

For θ in (39) we can simply choose the mode of the distribution $\mathcal{P}(\mathcal{D}|\theta)$, i.e., use $\mathcal{P}(y_x|\mathcal{D},\theta_{\mathrm{ML}})$ in making prediction where $\theta_{\mathrm{ML}} = \arg\max_{\theta}\mathcal{P}(\mathcal{D}|\theta)$. This method is usually referred to as Type II maximum likelihood. Note that this method is also equivalent to the MAP estimate of $\ln\theta$ with a uniform prior distribution on $\ln\theta$ that corresponds to a non-informative prior distribution $\mathcal{P}(\theta)$ (Berger, 1985).[7]

5 Numerical Experiments

In numerical experiments, the initial values of the hyperparameters are chosen as $\kappa = 1/d$ and $\kappa_b = 100.0$, where d is the input dimension. The initial value of κ_0 is chosen from $\{0.1, 1, 10, 100\}$, at which the gradient descent can start smoothly; usually it is 10. In Bayesian inference, we use the routine L-BFGS-B (Byrd et al., 1995) as the gradient-based optimization package, and start from the default initial

[7] In full Bayesian treatment, these hyperparameters θ must be integrated over θ space. Hybrid Monte Carlo (HMC) methods (Duane et al., 1987; Neal, 1996) can be adapted here to efficiently approximate the integral. However, we have not done it in the present work.

states mentioned above to infer optimal hyperparameters. $\mathcal{N}(\mu, \sigma^2)$ is used to denote a Gaussian distribution with mean μ and variance σ^2. We begin by showing the behavior of BTSVC on two simulated data sets. Then we report the training results on the benchmark data sets used by Rätsch et al. (2001). The computer we used for these numerical experiments is PIII 866 PC with 384MB RAM and the operating system is Windows 2000.[8]

5.1 Simulated Data 1

We generated 50 samples with positive label by randomly sampling in a Gaussian distribution $\mathcal{N}(-2, 1)$ and 50 samples with negative label in $\mathcal{N}(+2, 1)$ as the original case. To study the effect of outliers on BTSVC, we also created a second case where an extra sample with negative label at -2 was inserted as an outlier. We tried BTSVC with the Gaussian covariance function (9) and also SVC with Gaussian Kernel. For SVC, leave one out validation error was used to determine optimal hyperparameters, which are the regularization parameter C and the κ in the Gaussian kernel, $\exp\left(-\frac{\kappa}{2}\|x_i - x_j\|^2\right)$.[9] The initial search for optimal hyperparameters was done on a 7×7 coarse grid linearly spaced in the region $\{(\log_{10} C, \log_{10} \kappa) | 0 \leq \log_{10} C \leq 3, -3 \leq \log_{10} \kappa \leq 0\}$, followed by a fine search on a 9×9 uniform grid linearly spaced by 0.1 in the $(\log_{10} C, \log_{10} \kappa)$ space. For BTSVC, the initial value of κ_0 was set at 0.1 and Bayesian inference was used to find the optimal hyperparameters. Their final hyperparameter settings are recorded in Table 1. Comparing the results of BTSVC on the two cases, we find that the effect of the outlier can be reduced by decreasing the hyperparameter κ_0. Moreover, the increase on κ

[8]The program we used in the experiments is available at http://guppy.mpe.nus.edu.sg/∼mpessk/btsvc/bisvm.zip, and the simulated data can be accessed from http://guppy.mpe.nus.edu.sg/∼mpessk/btsvc/simu.zip.

[9]When two sets of hyperparameters yield same leave one out validation error, we prefer the set with smaller number of SVs.

in Gaussian kernel narrows the kernel shape that restricts the influence of the outlier to a local region. The discriminant functions of BTSVC and SVC are compared in Figure 2(b) and 2(d) for the two cases. In both cases, the region $\{x : f(x) > 0\}$ is quite similar for both BTSVC and SVC. In the probabilistic class prediction as given in Figure 2(c), the regions $\{x : \mathcal{P}(y_x = +1|x) > 0.5\}$ are almost same for both cases. In the outlier case of BTSVC, several patterns around the outlier turn out to be SVs, causing the error bar to reduce drastically. Through the reduction on the error bar, these SVs contribute more confidence on class predictions in the region around the outlier. This is another undesirable feature of our model associated with the presence of outliers. However, there is one positive aspect. Note from Figure 2(c) that, among all training samples of class -1, the outlier at -2 gets the lowest value for class -1 probability; this property can help to identify the outlier. Once outliers are identified and removed BTSVC works much better.

5.2 Simulated Data 2

We compared the negative log-likelihood and test error with other well-known probabilistic methods on a two dimensional simulated data set. The samples with positive label were generated by randomly sampling in a two-dimensional Gaussian distribution $\mathcal{N}\big((-2, 0), \mathrm{diag}\{1, 2\}\big)$, while the samples with negative label were generated by sampling in $\mathcal{N}\big((+2, 0), \mathrm{diag}\{2, 1\}\big)$. The data set is composed of 1000 training samples and 20002 test samples. The negative log-likelihood on test set and test error rate are recorded in Table 2, together with the results of other probabilistic approaches that includes optimal Bayes classifier (Duda et al., 2001) using the true generative model, Bayes classifier using the generative model estimated from training data, kernel logistic regression (Keerthi et al., 2002), probabilistic output of classical SVC (Platt, 2000), standard Gaussian processes for classification (GPC) (Williams and Barber, 1998).[10] BTSVC and GPC yield quite similar test error as we expect since both use the Laplacian approximation in Bayesian approach and the difference

[10]The results of kernel logistic regression and probabilistic output of standard SVC are cited from Keerthi et al. (2002), where 5-fold cross validation was used for model selection.

only lies in the loss function used. Compared with kernel logistic regression, BTSVC yields lower error rate, but quite similar likelihood evaluation. A visual comparison with Bayes classifier and GPC is given in Figure 3. The predictive likelihood of BTSVC is slightly conservative due to the broad error bar in the regions away from the SVs.

In the next experiment, we compared the generalization performance and the computational cost of standard SVC and BTSVC on different size of the two dimensional simulated data. The size of training data set ranged from 10 to 1000. The set of 20002 test samples was used as the common test data for all training data sets. At each size, we repeated the experiment 20 times to reduce the randomness in training data generation. If the training data size is less than 100, leave one out validation error was used to determine optimal hyperparameters for SVC, otherwise 10-fold cross validation was used. The searching method we used was same as that described in Section 5.1, and the test error was obtained using the optimal hyperparameters. The comparison of generalization performance is given in Figure 4. BTSVC and GPC yield better and more stable generalization performance than SVC, especially when the training data size is small. Clearly, when the number of training samples is small, the Bayesian approaches are very much superior.

We present the CPU times used by the three algorithms for solution evaluation at one hyperparameter set, separately in the three lower graphs of Figure 4. The CPU times of BTSVC and GPC include the cost for the gradient evaluation. From the scaling result in these graphs, we find that each evaluation in GPC consumes more CPU time than BTSVC, and BTSVC consumes comparable CPU time as that required by SVC for solving its quadratic programming problem. However, SVC with cross validation requires hundreds of evaluations (this depends on the grid placed in hyperparameter space for searching and the number of folds we use in cross validation), while for BTSVC and GPC, the number of evaluations depends on the iterations required by the optimization method to converge, which is usually about 20. Thus, the overall CPU time used by BTSVC to find the optimal hyperparameters is much less than that used by SVC when the number of SVs of BTSVC is less than one thousand. However, as the number of training samples increases, BTSVC may get more SVs and then the evaluation of the gradient could become a very expensive step. In such a situation, SVC will be much more efficient than BTSVC.

For large data sets with high noise level, we cannot obtain desirable sparseness in the BTSVC solution due to the effect of outliers. However, when the noise level is not high, the sparseness property allows BTSVC to handle reasonably large size data sets that GPC cannot handle.

5.3 Benchmark Data

We also carried out Bayesian inference with Gaussian covariance function (9) on the benchmark data sets used by Rätsch et al. (2001).[11] We report the training results of BTSVC on these data sets in Table 3. The optimal hyperparameters used throughout the training on all partitions of that data set were determined by the average results of Bayesian inference on the first five partitions. We choose optimal hyperparameters in this way for a fair comparison with the results of SVC reported by Rätsch et al. (2001). A paired t-test is carried out to measure how likely it would be to obtain the observed t-statistic under the null hypothesis that there is no difference on test error between BTSVC and SVC. The t-statistic is evaluated by $t = \bar{e} \left(\frac{\sum_{i=1}^{n} (e_i - \bar{e})^2}{n(n-1)} \right)^{-1/2}$ where $\bar{e} = \frac{1}{n} \sum_{i=1}^{n} e_i$ and e_i denotes the test error difference between the two methods on the i-th partition. The p-value, i.e. the probability of observing the given result by chance given that the null hypothesis is true, is recorded in the last column of Table 3. A small p-value implies the difference is significant. BTSVC and SVC tie on the overall performance. Thus, the generalization capability of our Bayesian approach is very competitive.

We also carried out the training results of standard Gaussian processes for classification (GPC) (Williams and Barber, 1998),[12] to compare the generalization capability and computational cost with BTSVC, which can be taken as a comparison between

[11] These 100-partition benchmark data sets (only 20 partitions available for Image and Splice) and related experimental results reported by Rätsch et al. (2001) can be accessed from http://www.first.gmd.de/∼raetsch/data/benchmarks.htm.

[12] The source code for GPC we used is available at http://guppy.mpe.nus.edu.sg/∼mpessk/btsvc/gpc.zip, in which convex programming is used to find the MAP estimate on function values and Type II maximum likelihood

logistic loss function and trigonometric loss function. The optimal hyperparameters were determined by Bayesian inference, which was carried out independently on every partition. Their results are recorded in Table 4. The overall generalization performance of BTSVC closely matches GPC. Notice that BTSVC requires quite less computational cost on large data sets.

The correlation between evidence and generalization performance (measured by test error rate) in BTSVC can be seen from their contour graphs in hyperparameter space on the first partition of Banana and Waveform data sets in Figure 5.

For the next experiment, we chose the Image and Splice data, which have many input variables, to carry out feature selection with ARD Gaussian covariance function (34). The inputs $\{x_i\}$ in the training data were normalized to zero mean and unit variance dimension-wise and the initial values for all ARD parameters were chosen as $1/d$ where d is the input dimension. In Table 5, BTSVC using ARD Gaussian covariance function improves generalization performance from 12.35% to 5.29% on the Splice data sets. From the optimal ARD parameters, we find that only the $28^{\text{th}} - 34^{\text{th}}$ input dimensions are significantly relevant in the whole 60 dimensions. Thus, we create reduced Splice data sets by keeping the 7 relevant dimensions only. On the reduced data sets, both Gaussian (in Table 4) and ARD Gaussian kernel can still yield competitive performance. Based on these numerical experiments, we find that both BTSVC and GPC have the capacity to determine the relevant inputs and hence improve generalization. BTSVC has the additional advantage that it requires less overhead than GPC on large data sets.

6 Conclusion

In this paper, we proposed a Bayesian support vector classifier by introducing trigonometric likelihood function. In the probabilistic framework of stationary Gaussian processes, various computational procedures are provided for the MAP estimate and the evidence of the hyperparameters. Model adaptation and ARD feature selection are implemented intrinsically in hyperparameter inference. Furthermore, the sparse-

with Laplacian approximation is used to tune hyperparameters.

ness property associated with the new likelihood function reduces the computational cost significantly. Another benefit is the availability of class probabilities in making predictions. The results in numerical experiments verify that the generalization capability is excellent and that it is possible to tackle reasonably large data sets with moderate noise level using this approach.

Acknowledgements

Wei Chu gratefully acknowledges the financial support provided by the National University of Singapore through Research Scholarship. The reviewers' thoughtful comments are gratefully appreciated.

Appendix: Trigonometric Support Vector Classifier

In a reproducing kernel Hilbert space (RKHS), trigonometric support vector classifier (TSVC) takes the form of $f(x) = \langle \mathbf{w} \cdot \phi(x) \rangle + b$, where b is known as the bias, $\phi(\cdot)$ is the mapping function, and the dot product $\langle \phi(x_i) \cdot \phi(x_j) \rangle$ is also the reproducing kernel $K(x_i, x_j)$ of the RKHS. The optimal classifier is constructed by minimizing the following regularized functional

$$\mathcal{R}(\mathbf{w}, b) = \sum_{i=1}^{n} \ell_t(y_{x_i} \cdot f_{x_i}) + \frac{1}{2}\|\mathbf{w}\|^2 \tag{40}$$

where $\|\mathbf{w}\|^2$ is a norm in the RKHS and $\ell_t(\cdot)$ denotes the trigonometric loss function. By introducing a slack variables $\xi_i \geq 1 - y_{x_i} \cdot (\langle \mathbf{w} \cdot \phi(x_i) \rangle + b)$, the minimization problem (40) can be rewritten as

$$\min_{\mathbf{w},b,\xi} \mathcal{R}(\mathbf{w}, b, \xi) = 2 \sum_{i=1}^{n} \ln \sec \left(\frac{\pi}{4} \xi_i \right) + \frac{1}{2}\|\mathbf{w}\|^2 \tag{41}$$

subject to $y_{x_i} \cdot (\langle \mathbf{w} \cdot \phi(x_i) \rangle + b) \geq 1 - \xi_i$ and $0 \leq \xi_i < 2$, $\forall i$, which is referred as the *primal* problem. Let $\alpha_i \geq 0$ and $\gamma_i \geq 0$ be the corresponding Lagrange multipliers for the inequalities in the *primal* problem. The KKT conditions for the *primal* problem

(41) are

$$\mathbf{w} = \sum_{i=1}^{n} y_{x_i}\alpha_i\phi(x_i);$$

$$\sum_{i=1}^{n} y_{x_i}\alpha_i = 0;$$

$$\frac{\pi}{2}\tan\left(\frac{\pi}{4}\xi_i\right) = \alpha_i + \gamma_i, \forall i. \tag{42}$$

Notice that the implicit constraint $\xi_i < 2$ has been taken into account automatically in (42). Following the analogous arguments in Section 3.2, we can derive the *dual* problem as

$$\begin{aligned}\min_{\alpha} \mathcal{R}(\alpha) \quad = \quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(y_{x_i}\alpha_i)(y_{x_j}\alpha_j)K(x_i,x_j) - \sum_{i=1}^{n}\alpha_i \\ & + \sum_{i=1}^{n}\left[\frac{4}{\pi}\alpha_i \arctan\left(\frac{2\alpha_i}{\pi}\right) - \ln\left(1+\left(\frac{2\alpha_i}{\pi}\right)^2\right)\right]\end{aligned} \tag{43}$$

subject to $\sum_{i=1}^{n} y_{x_i}\alpha_i = 0$ and $\alpha_i \geq 0\ \forall i$.

Comparing with BTSVC, the only difference lies in the existence of the equality constraint $\sum_{i=1}^{n} y_{x_i}\alpha_i = 0$ in TSVC. The popular SMO algorithm (Platt, 1999; Keerthi et al., 2001) can be adapted to find the solution. The classifier is obtained as $f(x) = \langle\sum_{i=1}^{n} y_{x_i}\alpha_i\phi(x_i)\cdot\phi(x)\rangle + b = \sum_{i=1}^{n} y_{x_i}\alpha_i K(x_i,x) + b$ using the optimal solution of the *dual* problem (43). Cross validation can be used to choose the optimal parameters for the kernel function.

References

Berger, J. O. *Statistical Decision Theory and* B*ayesian Analysis*. Springer-Verlag, New York, second edition, 1985.

Bishop, C. M. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

Burges, C. J. C. A tutorial on support vector machines for pattern recogintion. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

Byrd, R. H., P. Lu, and J. Nocedal. A limited memory algorithm for bound con-

strained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16 (5):1190–1208, 1995.

Duane, S., A. D. Kennedy, and B. J. Pendleton. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

Duda, R. O., P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, New York, second edition, 2001.

Evgeniou, T., M. Pontil, and T. Poggio. A unified framework for regularization networks and support vector machines. A.I. Memo 1654, Massachusette Institute of Technology, 1999.

Fletcher, R. *Practical methods of optimization*. John Wiley and Sons, 1987.

Keerthi, S. S., K. Duan, S. K. Shevade, and A. N. Poo. A fast dual algarithm for kernel logistic regression. *Proceeding of the 19th International Conference on Machine Learning*, 33:82–95, 2002.

Keerthi, S. S., S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, March 2001.

Kwok, J. T. The evidence framework applied to support vector machines. *IEEE Transactions on Neural Networks*, 11(5):1162–1173, 2000.

Larsson, T., M. Patriksson, and A. Strömberg. Ergodic, primal convergence in dual subgradient schemes for convex programming. *Math. Program*, 86:283312, 1999.

MacKay, D. J. C. A practical Bayesian framework for back propagation networks. *Neural Computation*, 4(3):448–472, 1992.

MacKay, D. J. C. Bayesian methods for backpropagation networks. *Models of Neural Networks III*, pages 211–254, 1994.

Neal, R. M. B*ayesian Learning for Neural Networks*. Lecture Notes in Statistics. Springer, 1996.

Platt, J. C. Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, 1999.

Platt, J. C. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Smola, A. J., P. L. Bartlett, B. Schölkopf, and D. Schuurnabs, editors, *Advances in Large Margin Classifier*, pages 61–73. MIT Press, 2000.

Rätsch, G., T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

Sollich, P. Bayesian methods for support vector machines: Evidence and predictive class probabilities. *Machine Learning*, 46:21–52, 2002.

Vanderbei, R. J. *Linear Programming: Foundations and Extensions*, volume 37 of *International Series in Operations Research and Management Science*. Kluwer Academic, Boston, 2nd edition, June 2001.

Vapnik, V. N. *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

Wahba, G. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, 1990.

Williams, C. K. I. and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

Table 1: The optimal hyperparameters in Gaussian covariance function (9) of BTSVC after hyperparameter inference, along with the model parameters of standard SVC with Gaussian kernel after leave one out cross validation on the one-dimensional simulated data set. The Evidence, $-\ln \mathcal{P}(\mathcal{D}|\theta)$, is evaluated using (32). SVs denotes the number of SVs. C denotes the regularization parameter in SVC.

	BTSVC					SVC		
Data set	κ_0	κ	κ_b	SVs	Evidence	C	κ	SVs
Original Case	7.485	0.194	0.565	9	5.46	7.943	0.159	6
Outlier Case	0.776	0.792	0.113	74	15.97	158.489	0.0158	8

Table 2: Negative log-likelihood on test set (NLL) and the error rate on test set (ERR) for optimal Bayes classifier (Optimal), Bayes classifier (Bayes), kernel logistic regression (Klogr), probabilistic output of classical support vector classifier (SVC), standard Gaussian processes for classification (GPC) and Bayesian trigonometric support vector classifier (BTSVC) on the two dimensional simulated data set.

	Optimal	Bayes	Klogr	SVC	GPC	BTSVC
NLL	2532.5	2559.2	2663.4	2703.5	2570.3	2665.7
ERR	0.0490	0.0495	0.0502	0.0507	0.0496	0.0496

Table 3: Training results of BTSVC with Gaussian covariance function (9) on the 100-partition benchmark data sets. d denotes the input dimension, n is the size of training data and m is the size of test data. κ_0, κ and κ_b denote the average result of BTSVC on the first five partitions. RATE denotes the test error rate (in percentage) averaged over all partitions of that data set together with the standard deviation; and for comparison purpose, we cite the test error rate of classical SVC with Gaussian kernel reported by Rätsch et al. (2001) in the column SVC, and then compute the p-values for the paired $t-$test on error rate difference. We use bold face to indicate the cases in which the indicated element is significantly better; a p-value threshold of 0.01 was used to decide this.

Data set	d	n	m	κ_0	κ	κ_b	RATE	SVC	p-value
Banana	2	400	4900	2.308	1.425	0.349	**10.39±0.50**	11.53±0.66	7.69×10^{-30}
Breast	9	200	77	0.172	0.115	0.00343	25.70±4.46	26.04±4.74	0.214
Diabetis	8	468	300	0.386	0.0606	15.638	**23.13±1.75**	23.53±1.73	2.95×10^{-4}
Flare	9	666	400	0.802	0.316	0.0969	34.26±1.75	**32.43±1.82**	1.01×10^{-17}
German	20	700	300	0.339	0.0625	11.362	23.37±2.28	23.61±2.07	0.0583
Heart	13	170	100	3.787	0.00731	9.222	16.33±2.78	15.95±3.26	0.0404
Image	18	1300	1010	87.953	0.0428	95.847	3.50±0.62	**2.96±0.60**	9.74×10^{-5}
Ringnorm	20	400	7000	0.978	0.0502	102.126	1.99±0.26	**1.66±0.12**	2.75×10^{-22}
Splice	60	1000	2175	3.591	0.00601	121.208	12.36±0.72	**10.88±0.66**	2.54×10^{-11}
Thyroid	5	140	75	66.920	0.132	96.360	**3.95±2.07**	4.80±2.19	3.41×10^{-8}
Titanic	3	150	2051	0.391	0.966	44.536	22.51±1.01	22.42±1.02	0.280
Twonorm	20	400	7000	18.658	0.00426	94.741	**2.90±0.27**	2.96±0.23	3.26×10^{-3}
Waveform	21	400	4600	1.310	0.0393	111.23	9.94±0.42	9.88±0.43	0.196

Table 4: Training results of BTSVC and GPC with Gaussian covariance function (9) on the 100-partition benchmark data sets. Splice* denotes the reduced Splice data sets, and SVs denotes the number of the SVs of BTSVC. RATE denotes the test error rate of BTSVC in percent averaged over all partitions of that data set together with the standard deviation, and GPC-RATE is the corresponding element of GPC. TIME denotes the average CPU time in seconds consumed by BTSVC for training on one partition, and GPC-TIME is the corresponding value of GPC. The p-value is for the paired $t-$test on error rate. We use the bold face to indicate the cases in which the indicated element is significantly better; a p-value threshold of 0.01 was used to decide this.

Data set	n	SVs	TIME	GPC-TIME	RATE	GPC-RATE	p-value
Banana	400	252.9±27.3	9.34±2.34	18.04±6.32	10.44±0.48	10.47±0.46	0.216
Breast	200	199.9±0.4	3.21±0.68	1.53±0.32	26.53±4.60	26.79±4.50	0.200
Diabetis	468	454.0±7.0	27.24±8.93	12.67±1.17	**23.21±1.77**	23.71±2.08	3.75×10^{-7}
Flare	666	646.5±14.4	71.61±21.05	47.87±14.92	34.39±1.81	34.22±1.81	0.0506
German	700	682.7±13.8	95.71±34.76	57.78±13.93	23.48±2.11	23.81±2.17	0.0115
Heart	170	149.1±8.6	1.77±0.58	2.39±0.65	**16.34±2.90**	17.19±3.23	1.22×10^{-4}
Image	1300	357.1±32.3	96.05±21.43	997.78±158.56	3.58±0.67	3.39±0.81	0.299
Ringnorm	400	188.8±9.0	2.88±1.19	18.79±9.94	1.99±0.26	**1.61±0.13**	7.36×10^{-39}
Splice	1000	713.8±21.4	261.43±60.39	519.52±65.21	12.35±0.75	**11.30±0.77**	3.23×10^{-11}
Splice*	1000	511.4±52.2	52.81±23.49	271.38±59.62	5.85±0.53	**5.59±0.46**	1.16×10^{-3}
Thyroid	140	30.6±8.3	0.28±0.13	1.56±0.32	**4.32±2.09**	4.80±1.94	4.41×10^{-4}
Titanic	150	149.8±1.5	1.32±0.38	0.90±0.22	22.73±1.43	**22.50±1.54**	5.12×10^{-3}
Twonorm	400	96.0±18.3	2.16±0.95	23.71±6.93	2.85±0.29	2.89±0.27	0.016
Waveform	400	190.7±22.3	4.38±1.77	30.92±6.24	10.11±0.45	10.06±0.47	0.0888

Table 5: Training results of BTSVC and GPC with ARD Gaussian kernel (34) on the Image and Splice 20-partition data sets. Splice* denotes the reduced Splice data sets, and SVs denotes the number of the SVs. RATE denotes the test error rate of BTSVC in percent averaged over all partitions of that data set together with the standard deviation, and GPC-RATE is the corresponding element of GPC. TIME denotes the average CPU time in seconds consumed by BTSVC for training on one partition, and GPC-TIME is the corresponding value of GPC. The p-value is for the paired $t-$test on error rate.

Data set	SVs	TIME	GPC-TIME	RATE	GPC-RATE	p-value
Image	379.5±53.7	133.71±86.89	1561.64±351.92	2.59±0.54	2.24±0.58	0.0287
Splice	598.1±74.4	811.89±574.48	1238.22±318.96	5.29±0.67	5.07±0.79	0.217
Splice*	491.6±37.3	48.43±21.32	498.04±247.84	5.71±0.59	5.59±0.55	0.303

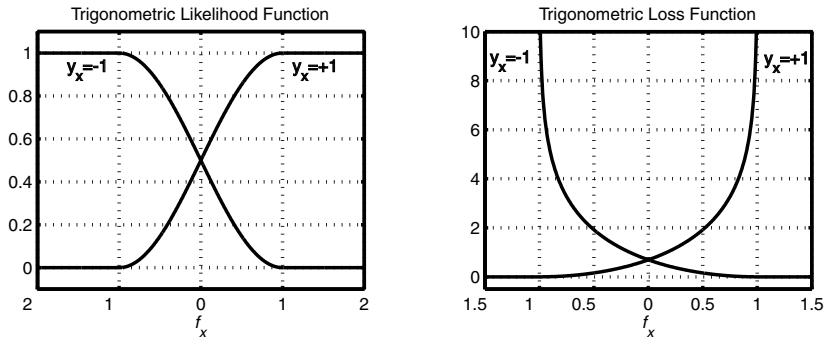

Figure 1: The graphs of trigonometric likelihood function and its loss function. The horizontal axis indicates the latent function f_x of the input vector x.

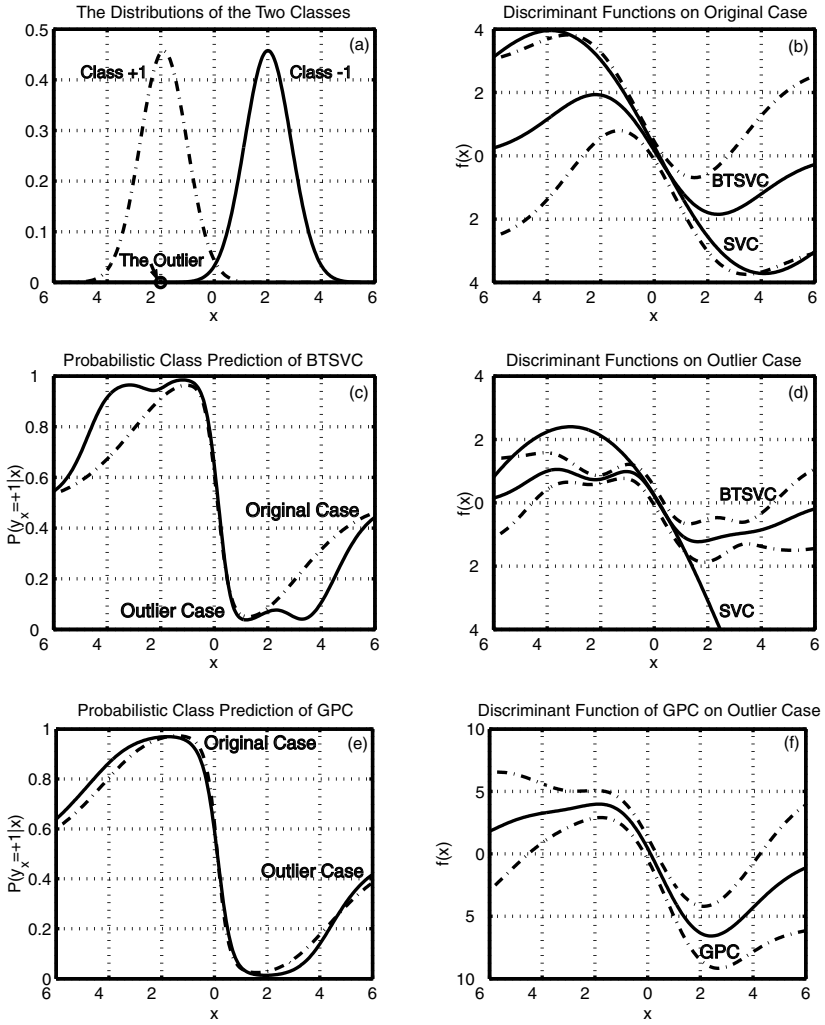

Figure 2: The training results of BTSVC on the one-dimensional simulated data, together with the results of SVC and GPC. In graph (a), the distributions of each class are presented as reference. In graph (c), we compare the probabilistic class prediction of BTSVC (39) on the original and outlier cases. In graph (e), we present the results of GPC on the two cases. In graphs (b) and (d), we plot the discriminant function of BTSVC, μ_t in (38), together with that of classical SVC for the two cases. The dotted curves indicate the error bars provided by BTSVC, i.e. $\mu_t \pm \sigma_t$. Leave one out cross validation was used to choose the optimal model parameters for SVC. In graph (f), we present the results of GPC on the outlier case as reference. The dotted curves indicate the error bars provided by GPC. Note that the error bars on $f(x)$ in (d) and (f) are not comparable, since the BTSVC and GPC have different relations linking $f(x)$ to the class probabilities.

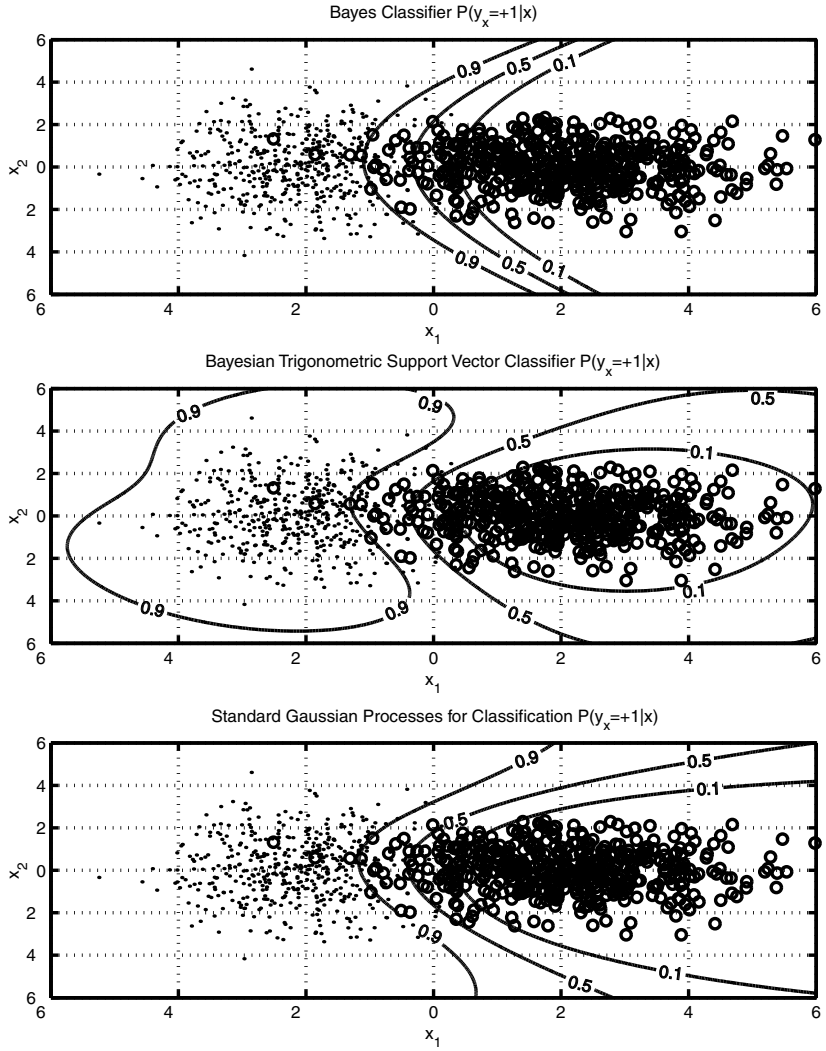

Figure 3: In the upper graph, the contour of the probabilistic output of Bayes classifier on the two dimensional simulated data set is presented. In the middle graph, the contour of probabilistic output of BTSVC is presented. In the lower graph, the contour of probabilistic output of standard Gaussian processes for classification is presented. The contours are indexed by $P(y_x = +1|x, \mathcal{D}, \theta)$.

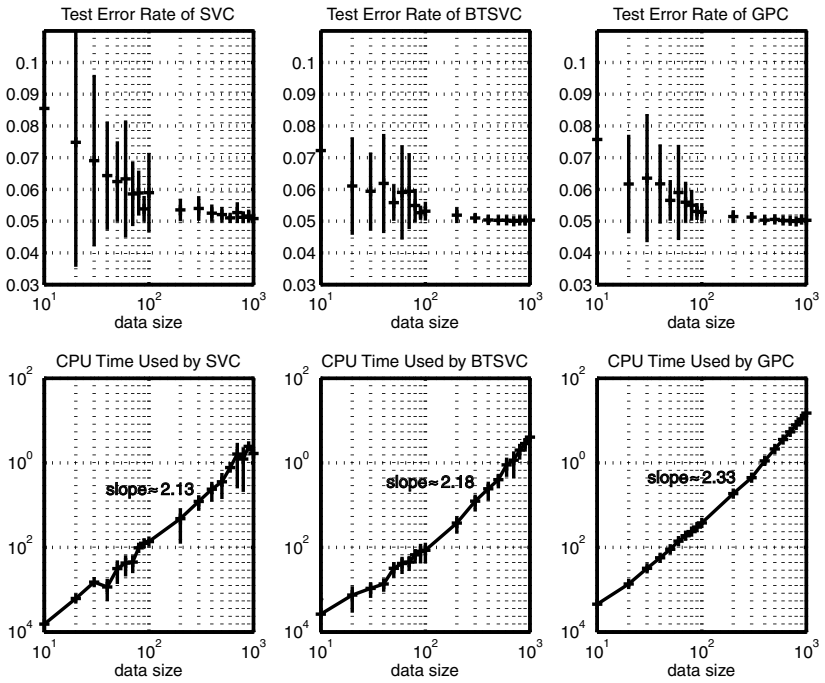

Figure 4: SVC, BTSVC and GPC on the two-dimensional simulated data sets at different size of training data. The test error rates at different sizes of training data are given in the three upper graphs separately. BTSVC and GPC used Bayesian inference with Laplacian approximation to tune hyperparameters while cross validation was used for SVC to choose optimal hyperparameters. In the left lower graph, the computational cost (CPU time in seconds) of SVC for training once on one fold is given. In the middle and right lower graph, we present the CPU time in seconds consumed by BTSVC and GPC for one evaluation on evidence and its gradient (including the convex programming) separately. The position of cross denotes the average value over 20 tries, and the vertical line indicates the standard deviation.

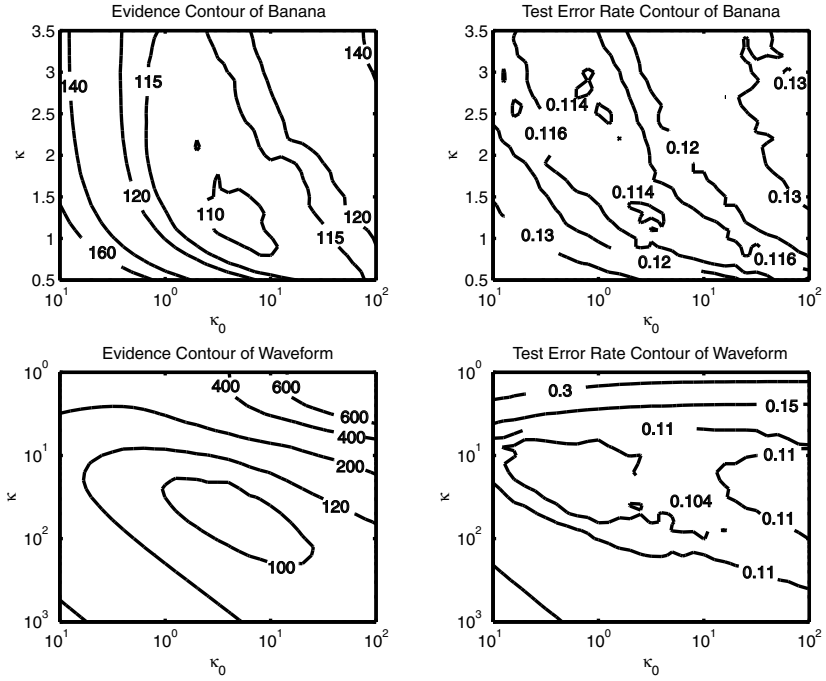

Figure 5: The contour graphs of evidence and testing error rate in hyperparameter space on the first fold of Banana and Waveform data sets. The horizontal axis indicates κ_0 and the vertical axis indicates κ. κ_b is fixed at 100. Note that the evidence is given by $-\ln \mathcal{P}(\mathcal{D}|\theta)$.