Bayesian Support Vector Regression Using a Unified Loss function

Wei Chu CHUWEI@GATSBY.UCL.AC.UK

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

In this paper, we use a unified loss function, called the soft insensitive loss function, for Bayesian support vector regression. We follow standard Gaussian processes for regression to set up the Bayesian framework, in which the unified loss function is used in the likelihood evaluation. Under this framework, the maximum a posteriori estimate of the function values corresponds to the solution of an extended support vector regression problem. The overall approach has the merits of support vector regression such as convex quadratic programming and sparsity in solution representation. It also has the advantages of Bayesian methods for model adaptation and error bars of its predictions. Experimental results on simulated and real-world data sets indicate that the approach works well even on large data sets.

Keywords: Bayesian Inference, Support Vector Regression, Gaussian Processes, Non-quadratic loss function, Automatic Relevance Determination, Model Selection

1 Introduction

The application of Bayesian techniques to neural networks was pioneered by Buntine and Weigend (1991), MacKay (1992) and Neal (1992). These works are reviewed in Bishop (1995), MacKay (1995) and Lampinen and Vehtari (2001). Unlike standard neural network design, the Bayesian approach considers probability distributions in the weight space of the network. Together with the observed data, prior distributions are converted to posterior distributions through the use of Bayes' theorem. Neal (1996) observed that a Gaussian prior for the weights approaches a Gaussian process for functions as the number of hidden units approaches infinity. Inspired by Neal's work, Williams and Rasmussen (1996) extended the use of Gaussian process prior to higher dimensional regression problems that have been traditionally tackled with other techniques, such as neural networks, decision trees etc, and good results have been obtained. Regression with Gaussian processes (GPR) is reviewed in Williams (1998). The important advantages of GPR models over other non-Bayesian models are the ability to infer hyperparameters

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

and the provision of confidence intervals of its predictions. The drawback of GPR models lies in the huge computational cost for large sets of data.

Support vector machines (SVM) for regression (SVR), as described by Vapnik (1995), exploit the idea of mapping input data into a high dimensional (often infinite) reproducing kernel Hilbert space (RKHS) where a linear regression is performed. The advantages of SVR are: the presence of a global minimum solution resulting from the minimization of a convex programming problem; relatively fast training speed; and sparseness in solution representation. The performance of SVR crucially depends on the shape of the kernel function and other hyperparameters that represent the characteristics of the noise distribution in the training data. Re-sampling approaches, such as cross-validation (Wahba, 1990), are commonly used in practice to decide values of these hyperparameters, but such approaches are very expensive when a large number of hyperparameters are involved. Typically, Bayesian methods are regarded as suitable tools to determine the values of these hyperparameters.

There is some literature on Bayesian interpretations of SVM. For classification, Kwok (2000) built up MacKay's evidence framework (MacKay, 1992) using a weight-space interpretation. Seeger (2000) presented a variational Bayesian method for model selection, and Sollich (2002) proposed Bayesian methods with normalized evidence and error bar. In SVM for regression (SVR), Law and Kwok (2001b) applied MacKay's Bayesian framework to SVR in the weight space. Gao et al. (2002) derived the evidence and error bar approximation for SVR along the way proposed by Sollich (2002). In these two approaches, the lack of smoothness of the ϵ-insensitive loss function (ϵ-ILF) in SVR may cause inaccuracy in evidence evaluation and inference. To improve the performance of Bayesian inference, we use a unified non-quadratic loss function for SVR, called the soft insensitive loss function (SILF). The SILF is C^1 smooth and possesses the main advantages of ϵ-ILF, such as insensitivity to outliers and sparseness in solution representation. We follow standard GPR to set up Bayesian framework, and then employ SILF in likelihood evaluation. Maximum a posteriori (MAP) estimate of the function values results in an extended SVR problem, so that quadratic programming can be employed to find the solution. Optimal hyperparameters can then be inferred by Bayesian techniques with the benefit of sparseness, and error bars of its predictions.

The important advantages of our Bayesian treatment on SVR using the SILF (BSVR) over classical SVR are: (1) the capability to systematically and efficiently infer optimal hyperparameters together with feature selection; and (2) the ability to compute predictive distribution using the probabilistic framework. Moreover, Bayesian model selection achieves quite better generalization performance on sparse training data sets than cross validation does. Compared with GPR, BSVR possesses: (1) the sparseness property that greatly reduces the computational cost in Bayesian inference and thus helps us to tackle large data sets, and (2) the insensitivity property to outliers that helps us to achieve robust performance on some real data sets.

The paper is organized as follows: in section 2 we review the standard framework of regression with Gaussian processes; in section 3 we propose SILF as a unified non-quadratic loss function, and describe some of its useful properties; in section 4 we employ SILF as the loss function in the MAP estimation of function values, and show that the associated optimization problem is a constrained convex quadratic programming problem; in section 5 we carry out hyperparameter inference by evidence maximization, and then intrinsically incorporate feature selection in the model adaptation; in section 6 we discuss the predictive distribution for test cases; in section 7 we carefully study our algorithm on simulated and real data sets, and also consistently compare our method with GPR and SVR for generalization performance and computational cost on benchmark data; we conclude the paper in section 8.

2 Bayesian Framework in Gaussian Processes

In regression problems, we are given a set of training data $\mathcal{D} = \{(x_i, y_i) | i = 1, \ldots, n, x_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}$ which is collected by randomly sampling a function f, defined on $\mathbb{R}^d$. As the measurements are usually corrupted by additive noise, training samples can be represented as

$$y_i = f(x_i) + \delta_i \quad i = 1, 2, \ldots, n \tag{1}$$

where the δ_i are independent, identically distributed (i.i.d.) random variables, whose distributions are usually unknown. Regression aims to infer the function f, or an estimate of it, from the finite data set $\mathcal{D}$. In the Bayesian approach, we regard the function f as the realization of a random field with a known prior probability. Let $\boldsymbol{f} = [f(x_1), f(x_2), \ldots, f(x_n)]^T$. The posterior probability of $\boldsymbol{f}$ given the training data $\mathcal{D}$ can then be derived by Bayes' theorem:

$$\mathcal{P}(\boldsymbol{f}|\mathcal{D}) = \frac{\mathcal{P}(\mathcal{D}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})}{\mathcal{P}(\mathcal{D})} \tag{2}$$

where $\mathcal{P}(\boldsymbol{f})$ is the prior probability of the random field and $\mathcal{P}(\mathcal{D}|\boldsymbol{f})$ is the conditional probability of the data $\mathcal{D}$ given the function values $\boldsymbol{f}$ which is exactly $\prod_{i=1}^{n} \mathcal{P}(y_i|f(x_i))$. Now we follow the standard Gaussian processes (Williams and Rasmussen, 1996; Williams, 1998) to describe a Bayesian framework.

2.1 Prior Probability

We assume that the collection of training data is the realization of random variables $f(x_i)$ in a zero mean stationary Gaussian process indexed by x_i. The Gaussian process is specified by the covariance matrix for the set of variables $\{f(x_i)\}$. The covariance between the outputs corresponding to inputs x_i and x_j can be defined as

$$Cov[f(x_i), f(x_j)] = Cov(x_i, x_j) = \kappa_0 \exp\left(-\frac{\kappa}{2}\sum_{l=1}^{d}(x_i^l - x_j^l)^2\right) + \kappa_b \tag{3}$$

where $\kappa > 0$, $\kappa_b > 0$ denotes the variance of the offset to the function $f(x)$, $\kappa_0 > 0$ denotes the average power of $f(x)$, and x^l denotes the l-th element of the input vector x. Such a covariance function expresses the idea that cases with nearby inputs have highly correlated outputs. Note that the first term in (3) is the Gaussian kernel in SVM, while the second term corresponds to the variance of the bias in classical SVR (Vapnik, 1995). Other kernel functions in SVM, such as polynomial kernel, spline kernel (Wahba, 1990), ANOVA decomposition kernel (Saunders et al., 1998) etc., or their combinations can also be used in covariance function, but we only focus on Gaussian kernel in the present work.

Thus, the prior probability of the functions is a multivariate Gaussian with zero mean and covariance matrix as follows

$$\mathcal{P}(\boldsymbol{f}) = \frac{1}{\mathcal{Z}_{\boldsymbol{f}}} \exp\left(-\frac{1}{2}\boldsymbol{f}^T \Sigma^{-1} \boldsymbol{f}\right) \tag{4}$$

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

2.2 Likelihood Function

The probability $\mathcal{P}(\mathcal{D}|\boldsymbol{f})$, known as likelihood, is essentially a model of the noise. If the additive noise δ_i in (1) is i.i.d. with probability distribution $\mathcal{P}(\delta_i)$, $\mathcal{P}(\mathcal{D}|\boldsymbol{f})$ can be evaluated by:

$$\mathcal{P}(\mathcal{D}|\boldsymbol{f}) = \prod_{i=1}^{n} \mathcal{P}(y_i - f(x_i)) = \prod_{i=1}^{n} \mathcal{P}(\delta_i) \tag{5}$$

Furthermore, $\mathcal{P}(\delta_i)$ is often assumed to be of the exponential form such that

$$\mathcal{P}(\delta_i) \propto \exp(-C \cdot \ell(\delta_i))$$

where $\ell(\cdot)$ is called the loss function and C is a parameter greater than zero. Thus, the likelihood function can also be expressed as

$$\mathcal{P}(\mathcal{D}|\boldsymbol{f}) \propto \exp\left(-C \cdot \sum_{i=1}^{n} \ell(y_i - f(x_i))\right) \tag{6}$$

Hence, the loss function characterizes the noise distribution which, together with the prior probability $\mathcal{P}(\boldsymbol{f})$, determines the posterior probability $\mathcal{P}(\boldsymbol{f}|\mathcal{D})$.

2.3 Posterior Probability

Based on Bayes' theorem (2), prior probability (4) and the likelihood (5), posterior probability of $\boldsymbol{f}$ can be written as

$$\mathcal{P}(\boldsymbol{f}|\mathcal{D}) = \frac{1}{\mathcal{Z}} \exp\left(-S(\boldsymbol{f})\right) \tag{7}$$

where $S(\boldsymbol{f}) = C\sum_{i=1}^{n} \ell\left(y_i - f(x_i)\right) + \frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f}$ and $\mathcal{Z} = \int \exp(-S(\boldsymbol{f}))d\boldsymbol{f}$. The maximum a posteriori (MAP) estimate of the function values is therefore the minimizer of the following optimization problem:[2]

$$\min_{\boldsymbol{f}} S(\boldsymbol{f}) = C\sum_{i=1}^{n} \ell\left(y_i - f(x_i)\right) + \frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f} \tag{8}$$

Let $\boldsymbol{f}_{\mathrm{MP}}$ be the optimal solution of (8). If the loss function in (8) is differentiable, the derivative of $S(\boldsymbol{f})$ with respect to $\boldsymbol{f}$ should be zero at $\boldsymbol{f}_{\mathrm{MP}}$, i.e.

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

[1] If the covariance is defined using (3), Σ is symmetric and positive definite if $\{x_i\}$ is a set of distinct points in $\mathbb{R}^d$ (Micchelli, 1986).

[2] $S(\boldsymbol{f})$ is a regularized functional. As for the connection to regularization theory, Evgeniou et al. (1999) have given a comprehensive discussion.

Let us define the following set of unknowns $w_i = -C\frac{\partial \ell(y_i - f(x_i))}{\partial f(x_i)}\Big|_{f(x_i)=f_{\mathrm{MP}}(x_i)}$ $\forall i$, and $\boldsymbol{w}$ as the column vector containing $\{w_i\}$. Then $\boldsymbol{f}_{\mathrm{MP}}$ can be written as:

$$\boldsymbol{f}_{\mathrm{MP}} = \Sigma \cdot \boldsymbol{w} \tag{9}$$

The elegant form of a minimizer of (8) is also known as the representer theorem (Kimeldorf and Wahba, 1971). A generalized representer theorem can be found in Schölkopf et al. (2001), in which the loss function is merely required to be any strictly monotonically increasing function $\ell : \mathbb{R} \to [0, +\infty)$.

2.4 Hyperparameter Evidence

The Bayesian framework we described above is conditional on the parameters in the prior distribution and the parameters in likelihood function, which can be collected as θ, the hyperparameter vector. The normalizing constant $\mathcal{P}(\mathcal{D})$ in (2), more exactly $\mathcal{P}(\mathcal{D}|\theta)$, is irrelevant to the inference of the functions $\boldsymbol{f}$, but it becomes important in hyperparameter inference, and it is known as the evidence of the hyperparameters θ (MacKay, 1992).

3 A Unified Non-quadratic Loss Function

There are several choices for the form of the loss function. In standard Gaussian processes for regression (GPR), Gaussian noise model is used in likelihood function (Williams and Rasmussen, 1996; Williams, 1998), which is of the form

$$\mathcal{P}_G(\delta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\delta^2}{2\sigma^2}\right). \tag{10}$$

where the parameter σ^2 is the noise variance and the corresponding loss function is the quadratic function $\ell(\delta) = \frac{1}{2}\delta^2$. This choice of the Gaussian likelihood, together with the Gaussian process prior for the functions $\boldsymbol{f}$, yields a posterior distribution of $\boldsymbol{f}$ that can be computed exactly using matrix operations in the GPR formulation. This is one reason why the Gaussian noise model is popularly used.

However, one of the potential difficulties of the quadratic loss function is that it receives large contributions from outliers. If there are long tails on the noise distributions then the solution can be dominated by a very small number of outliers, which is an undesirable result. Techniques that attempt to solve this problem are referred to as robust statistics (Huber, 1981). Non-quadratic loss functions have been introduced to reduce the sensitivity to the outliers. The three non-quadratic loss functions commonly used in regression problems are:

1. the Laplacian loss function defined as $\ell(\delta) = |\delta|$;
2. the Huber's loss function (Huber, 1981) defined as

$$\ell(\delta) = \begin{cases} \dfrac{\delta^2}{4\epsilon} & if \ \ |\delta| \le 2\epsilon \\ |\delta| - \epsilon & otherwise. \end{cases} \tag{11}$$

   where $\epsilon > 0$;

3. the ϵ-insensitive loss function (ϵ-ILF) (Vapnik, 1995),

$$\ell(\delta) = \begin{cases} 0 & if \ \ |\delta| \leq \epsilon \\ |\delta| - \epsilon & otherwise. \end{cases}$$

where $\epsilon > 0$.

From their definitions and Figure 1, the Huber's loss function and ϵ-ILF approach the Laplacian loss function as $\epsilon \rightarrow 0$. In addition, Laplacian loss function and ϵ-ILF are non-smooth, while Huber's loss function is a C^1 smooth function which can be thought of as a mixture between Gaussian and Laplacian loss function.

ϵ-ILF, used in SVR, is special in that it gives identical zero penalty to small noise values. Because of this, training samples with small noise that fall in this flat zero region are not involved in the representation of regression functions. This simplification of computational burden is usually referred to as the *sparseness* property. All the other loss functions mentioned above do not enjoy this property since they contribute a positive penalty to all noise values other than zero. On the other hand, quadratic and Huber's loss function are attractive because they are differentiable, a property that allows appropriate approximations to be used in the Bayesian approach. Based on these observations, we combine their desirable features and introduce a novel unified loss function called the soft insensitive loss function.

The soft insensitive loss function (SILF) is defined as:

$$\ell_{\epsilon,\beta}(\delta) = \begin{cases} -\delta - \epsilon & if \ \ \delta \in \Delta_{C^*} := \big(-\infty, -(1+\beta)\epsilon \big) \\ \dfrac{(\delta + (1-\beta)\epsilon)^2}{4\beta\epsilon} & if \ \ \delta \in \Delta_{M^*} := [-(1+\beta)\epsilon, -(1-\beta)\epsilon] \\ 0 & if \ \ \delta \in \Delta_0 := (-(1-\beta)\epsilon, (1-\beta)\epsilon) \\ \dfrac{(\delta - (1-\beta)\epsilon)^2}{4\beta\epsilon} & if \ \ \delta \in \Delta_M := [(1-\beta)\epsilon, (1+\beta)\epsilon] \\ \delta - \epsilon & if \ \ \delta \in \Delta_C := \big((1+\beta)\epsilon, +\infty \big) \end{cases} \tag{12}$$

where $0 < \beta \leq 1$ and $\epsilon > 0$. There is a profile of SILF as shown in Figure 2. The properties of SILF are entirely controlled by two parameters, β and ϵ. For a fixed ϵ, SILF approaches ϵ-ILF as $\beta \rightarrow 0$; on the other hand, it approaches the Huber's loss function as $\beta \rightarrow 1$. In addition, SILF becomes the Laplacian loss function as $\epsilon \rightarrow 0$. If ϵ is held at some large value and $\beta \rightarrow 1$, SILF approaches the quadratic loss function for all practical purposes.

The derivatives of the loss function are needed in Bayesian methods. The first order derivative of SILF can be written as

$$\frac{d\ell_{\epsilon,\beta}(\delta)}{d\delta} = \begin{cases} -1 & if \ \ \delta \in \Delta_{C^*} \\ \dfrac{\delta + (1-\beta)\epsilon}{2\beta\epsilon} & if \ \ \delta \in \Delta_{M^*} \\ 0 & if \ \ \delta \in \Delta_0 \\ \dfrac{\delta - (1-\beta)\epsilon}{2\beta\epsilon} & if \ \ \delta \in \Delta_M \\ 1 & if \ \ \delta \in \Delta_C \end{cases}$$

where $0 < \beta \leq 1$ and $\epsilon > 0$. The loss function is not twice continuously differentiable, but the second order derivative exists almost everywhere:

$$\frac{d^2\ell_{\epsilon,\beta}(\delta)}{d\delta^2} = \begin{cases} \dfrac{1}{2\beta\epsilon} & if \quad \delta \in \Delta_{M^*} \cup \Delta_M \\ 0 & otherwise \end{cases} \tag{13}$$

where $0 < \beta \leq 1$ and $\epsilon > 0$.

We now derive some of the properties of the noise model corresponding to SILF, as they are useful in the subsequent development. The density function of the additive noise in measurement corresponding to the choice of SILF is

$$\mathcal{P}_S(\delta) = \frac{1}{\mathcal{Z}_S} \exp\big(-C \cdot \ell_{\epsilon,\beta}(\delta)\big) \tag{14}$$

where $\dfrac{1}{\mathcal{Z}_S} = \displaystyle\int \exp\big(-C \cdot \ell_{\epsilon,\beta}(\delta)\big)\, d\delta$. It is possible to evaluate the integral and write $\mathcal{Z}_S$ as:

$$\mathcal{Z}_S = 2(1-\beta)\epsilon + 2\sqrt{\frac{\pi\beta\epsilon}{C}} \cdot \operatorname{erf}\Big(\sqrt{C\beta\epsilon}\Big) + \frac{2}{C}\exp\big(-C\beta\epsilon\big) \tag{15}$$

where $\operatorname{erf}(z) = \dfrac{2}{\sqrt{\pi}} \displaystyle\int_0^z \exp(-t^2)\, dt$. The mean of the noise is zero, and the variance of the noise σ_n^2 can be written as:

$$\begin{aligned} \sigma_n^2 = \; & \frac{2}{\mathcal{Z}_S}\Bigg\{\frac{(1-\beta)^3\epsilon^3}{3} + \sqrt{\frac{\pi\beta\epsilon}{C}}\left(\frac{2\beta\epsilon}{C} + (1-\beta)^2\epsilon^2\right)\operatorname{erf}\Big(\sqrt{\mathrm{C}\beta\epsilon}\Big) \\ & + \frac{4(1-\beta)\beta\epsilon^2}{C} + \left(\frac{\epsilon^2(1-\beta)^2}{C} + \frac{2\epsilon(1+\beta)}{C^2} + \frac{2}{C^3}\right)\exp(-C\beta\epsilon)\Bigg\} \end{aligned} \tag{16}$$

Remark 1 *We now give an interpretation for SILF, which is an extension of that given by Pontil et al. (1998) for ϵ-ILF. If we discard the popular assumption that the distribution of the noise variables δ_i is a zero-mean Gaussian, but assume that the noise variables δ_i have a Gaussian distribution $\mathcal{P}(\delta_i|\sigma_i, t_i)$ having its own standard deviation σ_i and its mean t_i that are i.i.d. random variables with density functions $\mu(\sigma_i)$ and $\lambda(t_i)$ respectively. Then we can compute the marginal of the noise probability by integrating over σ_i and t_i as follows:*

$$\mathcal{P}(\delta_i) = \int d\sigma_i \int dt_i \mathcal{P}(\delta_i|\sigma_i, t_i)\lambda(t_i)\mu(\sigma_i) \tag{17}$$

The probability (17) can also be evaluated in the form of loss function as (14). Under such settings, it is possible (Chu et al., 2001) to find a Rayleigh distribution on σ_i and a specific distribution on t_i, such that the evaluations of expression (14) and (17) are equivalent. Therefore, the use of SILF can also be explained as a general Gaussian noise model with the specific distribution on the mean and the standard deviation.

4 Support Vector Regression

We now describe the optimization problem (8) arising from the introduction of SILF (14) as the loss function. In this case, the MAP estimate of the function values is the minimizer of the following problem

$$\min_{\boldsymbol{f}} S(\boldsymbol{f}) = C\sum_{i=1}^{n} \ell_{\epsilon,\beta}(y_i - f(x_i)) + \frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f} \tag{18}$$

As usual, by introducing two slack variables ξ_i and ξ_i^*, (18) can be restated as the following equivalent optimization problem, which we refer to as the *primal* problem:

$$\min_{\boldsymbol{f},\boldsymbol{\xi},\boldsymbol{\xi}^*} S(\boldsymbol{f},\boldsymbol{\xi},\boldsymbol{\xi}^*) = C\sum_{i=1}^{n}(\psi(\xi_i) + \psi(\xi_i^*)) + \frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f} \tag{19}$$

subject to

$$\begin{cases} y_i - f(x_i) \leq (1-\beta)\epsilon + \xi_i \\ f(x_i) - y_i \leq (1-\beta)\epsilon + \xi_i^* \\ \xi_i \geq 0, \xi_i^* \geq 0 \quad \forall i \end{cases} \tag{20}$$

where

$$\psi(\pi) = \begin{cases} \frac{\pi^2}{4\beta\epsilon} & if \quad \pi \in [0, 2\beta\epsilon) \\ \pi - \beta\epsilon & if \quad \pi \in [2\beta\epsilon, +\infty) \end{cases} \tag{21}$$

Standard Lagrangian techniques (Fletcher, 1987) are used to derive the *dual* problem. Let $\alpha_i \geq 0$, $\alpha_i^* \geq 0$, $\gamma_i \geq 0$ and $\gamma_i \geq 0$ $\forall i$ be the corresponding Lagrange multipliers for the inequality in (20). The Lagrangian for the *primal* problem is:

$$\begin{aligned} L(\boldsymbol{f},\boldsymbol{\xi},\boldsymbol{\xi}^*;\boldsymbol{\alpha},\boldsymbol{\alpha}^*,\boldsymbol{\gamma},\boldsymbol{\gamma}^*) = C\sum_{i=1}^{n}(\psi(\xi_i) + \psi(\xi_i^*)) + \frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f} - \sum_{i=1}^{n}\gamma_i\xi_i - \sum_{i=1}^{n}\gamma_i^*\xi_i^* \\ -\sum_{i=1}^{n}\alpha_i(\xi_i + (1-\beta)\epsilon - y_i + f(x_i)) - \sum_{i=1}^{n}\alpha_i^*(\xi_i^* + (1-\beta)\epsilon + y_i - f(x_i)) \end{aligned} \tag{22}$$

The KKT conditions for the *primal* problem require

$$f(x_i) = \sum_{j=1}^{n}(\alpha_j - \alpha_j^*)Cov(x_i, x_j) \qquad \forall i \tag{23}$$

$$C\frac{\partial\psi(\xi_i)}{\partial\xi_i} = \alpha_i + \gamma_i \qquad \forall i \tag{24}$$

$$C\frac{\partial\psi(\xi_i^*)}{\partial\xi_i^*} = \alpha_i^* + \gamma_i^* \qquad \forall i \tag{25}$$

Based on the definition of $\psi(\cdot)$ given by (21) and the constraint conditions (24) and (25), the equality constraint on Lagrange multipliers can be explicitly written as

$$\alpha_i + \gamma_i = C\frac{\xi_i}{2\beta\epsilon} \text{ for } 0 \leq \xi_i < 2\beta\epsilon \text{ and } \alpha_i + \gamma_i = C \text{ for } \xi_i \geq 2\beta\epsilon \quad \forall i \tag{26}$$

$$\alpha_i^* + \gamma_i^* = C\frac{\xi_i^*}{2\beta\epsilon} \text{ for } 0 \leq \xi_i^* < 2\beta\epsilon \text{ and } \alpha_i^* + \gamma_i^* = C \text{ for } \xi_i^* \geq 2\beta\epsilon \quad \forall i \tag{27}$$

If we collect all terms involving ξ_i in the Lagrangian (22), we get $T_i = C\psi(\xi_i) - (\alpha_i + \gamma_i)\xi$. Using (21) and (26) we can rewrite T_i as

$$T_i = \begin{cases} -\dfrac{(\alpha_i + \gamma_i)^2\beta\epsilon}{C} & if \quad 0 \leq \alpha_i + \gamma_i < C \\ -C\beta\epsilon & if \quad \alpha_i + \gamma_i = C \end{cases} \tag{28}$$

Thus ξ_i can be eliminated if we set $T_i = -\frac{(\alpha_i+\gamma_i)^2\beta\epsilon}{C}$ and introduce the additional constraints, $0 \le \alpha_i + \gamma_i \le C$. The same arguments can be repeated for ξ_i^*. Then the *dual* problem becomes a maximization problem involving only the dual variables $\boldsymbol{\alpha}$, $\boldsymbol{\alpha}^*$, $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma}^*$:

$$\begin{aligned}
\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*,\boldsymbol{\gamma},\boldsymbol{\gamma}^*} S(\boldsymbol{\alpha},\boldsymbol{\alpha}^*,\boldsymbol{\gamma},\boldsymbol{\gamma}^*) &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)Cov(x_i,x_j) + \sum_{i=1}^{n}(\alpha_i-\alpha_i^*)y_i \\
&\quad -\sum_{i=1}^{n}(\alpha_i+\alpha_i^*)(1-\beta)\epsilon - \frac{\beta\epsilon}{C}\sum_{i=1}^{n}\left((\alpha_i+\gamma_i)^2+(\alpha_i^*+\gamma_i^*)^2\right)
\end{aligned} \tag{29}$$

subject to $\alpha_i \ge 0$, $\gamma_i \ge 0$, $\alpha_i^* \ge 0$, $\gamma_i^* \ge 0$, $0 \le \alpha_i + \gamma_i \le C$ and $0 \le \alpha_i^* + \gamma_i^* \le C$, $\forall i$. As the last term in (29) is the only one where γ_i and γ_i^* appear, (29) is maximal when $\gamma_i = 0$ and $\gamma_i^* = 0$ $\forall i$. Therefore, the *dual* problem can be finally simplified as

$$\begin{aligned}
\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} S(\boldsymbol{\alpha},\boldsymbol{\alpha}^*) &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)Cov(x_i,x_j) - \sum_{i=1}^{n}(\alpha_i-\alpha_i^*)y_i \\
&\quad +\sum_{i=1}^{n}(\alpha_i+\alpha_i^*)(1-\beta)\epsilon + \frac{\beta\epsilon}{C}\sum_{i=1}^{n}\left(\alpha_i^2+\alpha_i^{*2}\right)
\end{aligned} \tag{30}$$

subject to $0 \le \alpha_i \le C$ and $0 \le \alpha_i^* \le C$.

Obviously, the *dual* problem (30) is a convex quadratic programming problem. Matrix-based quadratic programming techniques that use the "chunking" idea can be used for its solutions (Vanderbei, 2001). Popular SMO algorithms for classical SVR (Smola and Schölkopf, 1998; Shevade et al., 2000) can also be adapted for its solution. For more details about the adaptation, refer to Chu (2003).

The optimal value of the primal variables $\boldsymbol{f}$ can be obtained from the solution of (30) as

$$\boldsymbol{f}_{\mathrm{MP}} = \Sigma \cdot (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) \tag{31}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T$ and $\boldsymbol{\alpha}^* = [\alpha_1^*, \alpha_2^*, \ldots, \alpha_n^*]^T$. This expression, which is consistent with (9), is the solution to MAP estimate of the function values $\boldsymbol{f}_{\mathrm{MP}}$ in the Gaussian processes.[3] At the optimal solution, the training samples (x_i, y_i) with associated $\alpha_i - \alpha_i^*$ satisfying $0 < |\alpha_i - \alpha_i^*| < C$ are usually called *off-bound* support vectors (SVs); the samples with $|\alpha_i - \alpha_i^*| = C$ are *on-bound* SVs, and the samples with $|\alpha_i - \alpha_i^*| = 0$ are non-SVs. From the definition of SILF (12) and the equality constraints (26) and (27), we notice that the noise δ_i in (1) associated with on-bound SVs should belong to $\Delta_{C^*} \cup \Delta_C$, while δ_i associated with off-bound SVs should belong to the region $\Delta_{M^*} \cup \Delta_M$.[4]

Remark 2 *From (13), the second derivative of $\ell_{\epsilon,\beta}(\delta_i)$ is not continuous at the boundary of $\Delta_{M^*} \cup \Delta_M$. The lack of C^2 continuity may have impact on the evaluation of the evidence $\mathcal{P}(\mathcal{D}|\theta)$ (to be discussed later in Section 5). However, it should be pointed out that the noise δ_i seldom falls on the boundary of $\Delta_{M^*} \cup \Delta_M$ exactly, since it is of low probability for a continuous random variable to be realized on some particular values.*

[3] In Gaussian processes, the most probable estimate and the MAP estimate are identical.

[4] Note that the region $\Delta_{M^*} \cup \Delta_M$ is crucially determined by the parameter β in the SILF (12).

4.1 General Formulation

Like SILF, the dual problem in (30) is a generalization of several SVR formulations. More exactly, when $\beta = 0$ (30) becomes the SVR formulation using ϵ-ILF; when $\beta = 1$, (30) becomes that when the Huber's loss function is used; and when $\beta = 0$ and $\epsilon = 0$, (30) becomes that for the case of the Laplacian loss function. Moreover, for the case of Gaussian noise model (10), the *dual* problem becomes

$$\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)Cov(x_i,x_j) - \sum_{i=1}^{n}(\alpha_i-\alpha_i^*)y_i + \frac{\sigma^2}{2}\sum_{i=1}^{n}\left(\alpha_i^2+\alpha_i^{*2}\right) \tag{32}$$

subject to $\alpha_i \geq 0$ and $\alpha_i^* \geq 0$ $\forall i$, where σ^2 is the variance of the additive Gaussian noise. The optimization problem (30) is equivalent to the general SVR (30) with $\beta = 1$ and $2\epsilon/C = \sigma^2$ provided that we keep upper bound C large enough to prevent any α_i and α_i^* from reaching the upper bound at the optimal solution. If we take the implicit constraint $\alpha_i \cdot \alpha_i^* = 0$ into account and then denote $\alpha_i - \alpha_i^*$ as ν_i, it is found that the formulation (32) corresponds to a much simpler case of

$$\min_{\boldsymbol{\nu}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\nu_i\nu_j Cov(x_i,x_j) - \sum_{i=1}^{n}\nu_i y_i + \frac{\sigma^2}{2}\sum_{i=1}^{n}\nu_i^2 \tag{33}$$

without any constraint. This is an unconstrained quadratic programming problem. The solution on small data sets can be found simply from a matrix inverse operation. For large data sets, conjugate gradient algorithm could be used (Luenberger, 1984). As for SMO algorithm design, see the ideas on LS-SVMs discussed by Keerthi and Shevade (2003).

5 Model Adaptation

The hyperparameter vector θ contains the parameters in the prior distribution and the parameters in the likelihood function, i.e., $\theta = \{C, \epsilon, \kappa, \kappa_b\}$.[5] For a given set of θ, the MAP estimate of the functions can be found from the solution of the optimization problem (18) in Section 4. Based on the MAP estimate $\boldsymbol{f}_{\mathrm{MP}}$, we show below how the optimal values of the hyperparameters are inferred.

5.1 Evidence Approximation

The optimal values of hyperparameters θ can be inferred by maximizing the posterior probability $\mathcal{P}(\theta|\mathcal{D})$:

$$\mathcal{P}(\theta|\mathcal{D}) = \frac{\mathcal{P}(\mathcal{D}|\theta)\mathcal{P}(\theta)}{\mathcal{P}(\mathcal{D})}$$

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

[5] Due to the redundancy with C and the correlation with κ, κ_0 is fixed at the variance of the targets $\{y_i\}$ instead of automatical tuning in the present work.

for $\mathcal{P}(\theta)$, i.e., $\mathcal{P}(\theta)$ is greatly insensitive to the values of θ. Therefore, the evidence $\mathcal{P}(\mathcal{D}|\theta)$ can be used to assign a preference to alternative values of the hyperparameters θ (MacKay, 1992). An explicit expression of the evidence $\mathcal{P}(\mathcal{D}|\theta)$ can be obtained as an integral over the f-space with a Taylor expansion at $\boldsymbol{f}_{\mathrm{MP}}$. Gradient-based optimization methods can then be used to infer the optimal hyperparameters that maximize this evidence function, given by

$$\mathcal{P}(\mathcal{D}|\theta) = \int \mathcal{P}(\mathcal{D}|\boldsymbol{f}, \theta)\mathcal{P}(\boldsymbol{f}|\theta)\, d\boldsymbol{f}. \tag{34}$$

Using the definitions of the prior probability (4) and the likelihood (5) with SILF (14), the evidence (34) can be written as

$$\mathcal{P}(\mathcal{D}|\theta) = \mathcal{Z}_{\boldsymbol{f}}^{-1}\mathcal{Z}_S^{-n} \int \exp\left(-S(\boldsymbol{f})\right)\, d\boldsymbol{f}. \tag{35}$$

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. The first order derivative with respect to $\boldsymbol{f}$ at the most probable point $\boldsymbol{f}$ is zero. The second order derivative exists everywhere except the boundary of the region $\Delta_M \cup \Delta_M^*$. As pointed out in Remark 2, the probability that a sample exactly falls on the boundary is little. Thus it is quite alright to use the second order approximation

$$S(\boldsymbol{f}) \approx S(\boldsymbol{f}_{\mathrm{MP}}) + \frac{1}{2}(\boldsymbol{f} - \boldsymbol{f}_{\mathrm{MP}})^T \cdot \left.\frac{\partial^2 S(\boldsymbol{f})}{\partial \boldsymbol{f} \partial \boldsymbol{f}^T}\right|_{\boldsymbol{f}=\boldsymbol{f}_{\mathrm{MP}}} \cdot (\boldsymbol{f} - \boldsymbol{f}_{\mathrm{MP}}) \tag{36}$$

where $\left.\frac{\partial^2 S(\boldsymbol{f})}{\partial \boldsymbol{f} \partial \boldsymbol{f}^T}\right|_{\boldsymbol{f}=\boldsymbol{f}_{\mathrm{MP}}} = \Sigma^{-1} + C \cdot \Lambda$ and Λ is a diagonal matrix with ii-th entry being $\frac{1}{2\beta\epsilon}$ if the corresponding training sample (x_i, y_i) is an *off-bound* SV, otherwise the entry is zero. Introducing (36) and $\mathcal{Z}_{\boldsymbol{f}}$ into (35), we get

$$\mathcal{P}(\mathcal{D}|\theta) = \exp\left(-S(\boldsymbol{f}_{\mathrm{MP}})\right) \cdot |\mathbf{I} + C \cdot \Sigma \cdot \Lambda|^{-\frac{1}{2}} \cdot \mathcal{Z}_S^{-n} \tag{37}$$

where $\mathbf{I}$ is a $n \times n$ identity matrix.

Notice that only a sub-matrix of Σ plays a role in the determinant $|\mathbf{I} + C \cdot \Sigma \cdot \Lambda|$ due to the sparseness of Λ. Let Σ_{M} be the $m \times m$ sub-matrix of Σ obtained by deleting all the rows and columns associated with the on-bound SVs and non-SVs, i.e., keeping the m off-bound SVs only. This fact, together with $\boldsymbol{f}_{\mathrm{MP}} = \Sigma \cdot (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)$ from (31), can be used to show that the negative log probability of data given hyperparameters is

$$-\ln \mathcal{P}(\mathcal{D}|\theta) = \frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \cdot \Sigma \cdot (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) + C \sum_{i=1}^{n} \ell_{\beta,\epsilon}(y_i - f_{\mathrm{MP}}(x_i)) + \frac{1}{2} \ln \left|\mathbf{I} + \frac{C}{2\beta\epsilon}\Sigma_{\mathrm{M}}\right| + n \ln \mathcal{Z}_S \tag{38}$$

where $\mathcal{Z}_S$ is defined by (15), $\mathbf{I}$ is a $m \times m$ identity matrix. The evidence evaluation (38) is a convenient yardstick for model selection.

The expression of (38) is then used for the determination of the best hyperparameter θ by finding the minimizer for $-\ln \mathcal{P}(\mathcal{D}|\theta)$. Note that the evidence depends on the set of *off-bound* SVs. This set will vary when the hyperparameters are changed. We assume that the set of *off-bound* SVs remains unchanged near the minimum of (38). In this region, the evidence is a smooth function of these hyperparameters. Gradient-based optimization methods could be used for the minimizer of (38). We usually collect $\{\ln C, \ln\epsilon, \ln\kappa_b, \ln\kappa\}$ as the set of variables to tune,[6] and the derivatives of $-\ln\mathcal{P}(\mathcal{D}|\theta)$ with respect to these variables are

$$\begin{aligned}\frac{\partial -\ln\mathcal{P}(\mathcal{D}|\theta)}{\partial\ln C} &= C\sum_{i=1}^{n}\ell_{\epsilon,\beta}(y_i - f_{\mathrm{MP}}(x_i)) + \frac{1}{2}\mathrm{trace}\left[\left(\frac{2\beta\epsilon}{C}\mathbf{I}+\Sigma_{\mathrm{M}}\right)^{-1}\Sigma_{\mathrm{M}}\right] \\ &\quad -\frac{n}{\mathcal{Z}_S}\left(\sqrt{\frac{\beta\epsilon\pi}{C}}\cdot\mathrm{erf}(\sqrt{C\beta\epsilon})+\frac{2}{C}\exp(-C\beta\epsilon)\right)\end{aligned} \tag{39}$$

$$\begin{aligned}\frac{\partial -\ln\mathcal{P}(\mathcal{D}|\theta)}{\partial\ln\epsilon} &= -C\left(\sum_{\delta_i\in\Delta_{M^*}\cup\Delta_M}\frac{\delta_i^2-(1-\beta)^2\epsilon^2}{4\beta\epsilon}+\sum_{\delta_i\in\Delta_{C^*}\cup\Delta_C}\epsilon\right) \\ &\quad -\frac{1}{2}\mathrm{trace}\left[\left(\frac{2\beta\epsilon}{C}\mathbf{I}+\Sigma_{\mathrm{M}}\right)^{-1}\Sigma_{\mathrm{M}}\right]+\frac{n}{\mathcal{Z}_S}\left(\sqrt{\frac{\beta\epsilon\pi}{C}}\cdot\mathrm{erf}(\sqrt{C\beta\epsilon})+2(1-\beta)\epsilon\right)\end{aligned} \tag{40}$$

$$\frac{\partial -\ln\mathcal{P}(\mathcal{D}|\theta)}{\partial\ln\kappa'} = \frac{\kappa'}{2}\mathrm{trace}\left[\left(\frac{2\beta\epsilon}{C}\mathbf{I}+\Sigma_{\mathrm{M}}\right)^{-1}\frac{\partial\Sigma_{\mathrm{M}}}{\partial\kappa'}\right]-\frac{\kappa'}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T\frac{\partial\Sigma}{\partial\kappa'}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \tag{41}$$

where $\kappa'\in\{\kappa_b,\kappa\}$, $\delta_i = y_i - f_{\mathrm{MP}}(x_i)$, and $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$ is the optimal solution of (30). Note that the non-SVs are not involved in these evaluations.[7]

5.2 Feature Selection

Feature selection is an essential part in regression modelling. Recently, Jebara and Jaakkola (2000) formalized a kind of feature weighting in maximum entropy discrimination framework, and Weston et al. (2001) introduced a method of feature selection for support vector machines by minimizing the bounds on the leave-one-out error.

MacKay (1994) and Neal (1996) proposed automatic relevance determination (ARD) 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 model adaptation, which reduces the influence of such a weight towards zero effectively. ARD can be directly embedded into the covariance function (3) as follows (Williams, 1998):

$$Cov[f(x_i),f(x_j)] = Cov(x_i,x_j) = \kappa_0\exp\left(-\frac{1}{2}\sum_{l=1}^{d}\kappa_l(x_i^l-x_j^l)^2\right)+\kappa_b \tag{42}$$

where $\kappa_l > 0$ is the ARD parameter that determines the relevance of the l-th input dimension to the prediction of the output variables. The derivatives of $-\ln\mathcal{P}(\mathcal{D}|\theta)$ with respect to the variables $\{\ln\kappa_l\}_{l=1}^d$ can be evaluated as in (41). The form of feature selection we use here results in a type of feature weighting.

It is possible that the optimization problem is stuck at local minima in the determination of θ. We minimize the impact of this problem by minimizing (38) several times starting from several different initial states, and choosing the one with the highest evidence as our preferred choice for θ. It is also possible to organize these candidates together as an expert committee to represent the predictive distribution that can reduce the uncertainty with respect to the hyperparameters.

[6] The definition of variables causes the optimization problem to be unconstrained.

[7] Refer to Chu (2003) for full details of the derivation.

5.3 Discussions

In classical GPR, the inversion of the full $n \times n$ matrix Σ has to be done for hyperparameter inference. In our approach, only the inversion of the $m \times m$ matrix Σ_{M}, corresponding to *off-bound* SVs, is required instead of the full matrix inverse. The non-SVs are not even involved in matrix multiplication and the future prediction. Usually, the *off-bound* SVs are small fraction of the whole training samples. As a result, it is possible to tackle reasonably large data sets with thousands of samples using our approach. For very large data sets, the size of the matrix Σ_{M} can still be large and the computation of its inverse can become the most time-consuming step. The parameter β can control the number of *off-bound* SVs. In the numerical experiments, we find that the choice of β has little influence on the training accuracy and the generalization capacity, but has a significant effect on the number of *off-bound* SVs and hence, the training time. As a practical strategy for tuning β, we can choose a suitable β to keep the number of *off-bound* SVs small for large data sets.[8] This can shorten training time greatly with no appreciable degradation in the generalization performance. Heuristically, we fix β at: 0.3 when the size of training data sets is less than 2000; 0.1 for $2000 \sim 4000$ samples; and, 0.05 for $4000 \sim 6000$ samples.[9]

Clearly, Our discussion above is not suitable to the case of classical SVR ($\beta = 0$), since in this case SILF becomes ϵ-ILF, which is not smooth. An approximate evaluation for the evidence in the case has been discussed by Gao et al. (2002), in which the (left/right) first order derivative at the insensitive tube is used in the evidence approximation.

Schölkopf and Smola (1998) proposed an interesting variant of SVR, known as $\nu-$SVR, in which the hyperparameter ϵ is optimized in the MAP estimate. Law and Kwok (2001a) applied the evidence framework (MacKay, 1992) to $\nu-$SVR with a particular prior for ϵ, but the dependency on ϵ makes the consequent evidence approximation intractable. Variational methods (Opper and Saad, 2001) might be used here to tackle the integral.

6 Error Bar in Prediction

In this section, we present error bars for predictions on new data points (MacKay, 1992; Bishop, 1995). This ability to provide error bars is one of the important advantages of the probabilistic approach over the usual deterministic approach to SVR.

Suppose a test case x is given for which the target t_x is unknown. The random variable $f(x)$ indexed by x along with the n random variables $\{f(x_i)\}$ in (4) have the joint multivariate Gaussian distribution,

$$\left[\begin{array}{c} \boldsymbol{f} \\ f(x) \end{array} \right] \sim \mathcal{N}\left(\left[\begin{array}{c} 0 \\ 0 \end{array} \right], \left[\begin{array}{cc} \Sigma & \boldsymbol{k} \\ \boldsymbol{k}^T & Cov(x,x) \end{array} \right] \right) \tag{43}$$

where $\boldsymbol{f}$ and Σ are defined as in (4), $\boldsymbol{k}^T = [Cov(x_1,x), Cov(x_2,x), \ldots, Cov(x_n,x)]$. The condi-

[8] Clearly, the number of *off-bound* SVs reduces, as $\beta \to 0$, to the number of *off-bound* SVs in the standard SVR ($\beta = 0$), but never below this number. The set of *off-bound* SVs in standard SVR is usually a small part of the training set.

[9] As for small size data sets, such as less than 100, we may set β at some large value, say $0.8 \sim 1.0$, to avoid the matrix Σ_{M} from shrinking.

tional distribution of $f(x)$ given $\boldsymbol{f}$ is a Gaussian,

$$\mathcal{P}(f(x)|\boldsymbol{f}) \propto \exp\left(-\frac{1}{2}\frac{(f(x) - \boldsymbol{f}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k})^2}{Cov(x,x) - \boldsymbol{k}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k}}\right) \tag{44}$$

where the mean is $E_{f(x)|\boldsymbol{f}}[f(x)] = \boldsymbol{f}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k}$ and the variance is $Var_{f(x)|\boldsymbol{f}}[f(x)] = Cov(x,x) - \boldsymbol{k}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k}$. At $\boldsymbol{f}_{\text{MP}}$, the mean of the predictive distribution for $f(x)$ is $\boldsymbol{f}_{\text{MP}}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k}$, where $\boldsymbol{f}_{\text{MP}}^T \cdot \Sigma^{-1}$ is just the Lagrange multipliers $(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T$ in the solution of (30).[10]

To make predictions with the optimal hyperparameters we have inferred, we need to compute the distribution $\mathcal{P}(f(x)|\mathcal{D})$ in order to erase the influence of the uncertainty in $\boldsymbol{f}$.[11] Formally, $\mathcal{P}(f(x)|\mathcal{D})$ can be found from

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

where $\mathcal{P}(f(x)|\boldsymbol{f})$ is given by (44) and $\mathcal{P}(\boldsymbol{f}|\mathcal{D})$ is given by (7). We replace $\boldsymbol{f} \cdot \Sigma^{-1}$ by its linear expansion around $\boldsymbol{f}_{\text{MP}}$ and use the approximation (36) for $S(\boldsymbol{f})$, the distribution $\mathcal{P}(f(x)|\mathcal{D})$ can be written as:

$$\begin{aligned}\mathcal{P}(f(x)|\mathcal{D}) \;\;\propto\;\; & \int \exp\left(-\frac{1}{2}\frac{(f(x) - \boldsymbol{f}_{\text{MP}}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k} - (\boldsymbol{f} - \boldsymbol{f}_{\text{MP}})^T \cdot \Sigma^{-1} \cdot \boldsymbol{k})^2}{Cov(x,x) - \boldsymbol{k}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k}}\right) \cdot \\ & \exp\left(-\frac{1}{2}(\boldsymbol{f} - \boldsymbol{f}_{\text{MP}})^T(\Sigma^{-1} + C \cdot \Lambda)(\boldsymbol{f} - \boldsymbol{f}_{\text{MP}})\right) d\boldsymbol{f}\end{aligned}$$

This expression can be simplified to a Gaussian distribution of the form:

$$\mathcal{P}(f(x)|\mathcal{D}) = \frac{1}{\sqrt{2\pi}\sigma_t} \exp\left(-\frac{(f(x) - (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \cdot \boldsymbol{k})^2}{2\sigma_t^2}\right) \tag{45}$$

where $\sigma_t^2 = Cov(x,x) - \boldsymbol{k}_{\text{M}}^T \cdot (\frac{2\beta\epsilon}{C}\mathbf{I} + \Sigma_{\text{M}})^{-1} \cdot \boldsymbol{k}_{\text{M}}$ and $\boldsymbol{k}_{\text{M}}$ is a sub-vector of $\boldsymbol{k}$ obtained by keeping the entries associated with the *off-bound* SVs.

The target t_x is a function of $f(x)$ and the noise δ as in (1), i.e. $t_x = f(x) + \delta$. As the noise is of zero mean, with variance σ_n^2 as given in (16), the variance of t_x is therefore $\sigma_t^2 + \sigma_n^2$.

7 Numerical Experiments

In the implementation of our Bayesian approach to support vector regression (BSVR), we used the routine L-BFGS-B (Byrd et al., 1995) as the gradient-based optimization package, and started from the initial values of the hyperparameters to infer the optimal ones.[12] We also

[10] The zero Lagrange multipliers in the solution of (30) associated with non-SVs are not at all involved in the prediction process.

[11] In a full Bayesian treatment, these hyperparameters θ must be integrated over θ-space. Hybrid Monte Carlo methods (Duane et al., 1987; Neal, 1992) can be adopted here to approximate the integral efficiently by using the gradients of $\mathcal{P}(\mathcal{D}|\theta)$ to choose search directions which favor regions of high posterior probability of θ.

[12] In numerical experiments, the initial values of the hyperparameters were usually chosen as $C = 1.0$, $\epsilon = 0.05$, $\kappa_b = 100.0$ and $\kappa = 0.5$. We suggest to try more starting points in practice, such as $C = 10.0$ or $\kappa = 1/d$ where d is the input dimension, and then choose the best model by the evidence.

implemented standard GPR (Williams, 1998) and classical SVR (Vapnik, 1995) for comparison purpose. For GPR, evidence maximization was implemented to choose the optimal hyperparameters using the routine L-BFGS-B. In the classical SVR, there are three tunable hyperparameters $\{C, \epsilon, \kappa\}$ in the case that the Gaussian covariance function (3) is used as the kernel function.[13] Due to the prohibitive computational cost for cross validation in three-dimensional hyperparameter space, we simply fix ϵ at a reasonable value and then search the corresponding optimal values for C and κ only. Five-fold cross validation was employed to determine their optimal values. The initial search was done on a 7×7 coarse grid linearly spaced in the region $\{(\log_{10} C, \log_{10} \kappa) | -0.5 \leq \log_{10} C \leq 2.5, -2.5 \leq \log_{10} \kappa \leq 0.5\}$, 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. This scheme requires 650 evaluations. In order to accelerate these experiments, we also cached the full covariance matrix in the implementation of GPR and SVR that requires $\mathcal{O}(n^2)$ memory, but we did not do that for BSVR. Average squared error (ASE) and average absolute error (AAE) are used as measures in prediction. Their definitions are

$$\text{ASE} = \frac{1}{m} \sum_{j=1}^{m} (y_j - f(x_j))^2 \quad \text{and} \quad \text{AAE} = \frac{1}{m} \sum_{j=1}^{m} |y_j - f(x_j)|$$

where m is the number of test cases, y_j is the target value for x_j and $f(x_j)$ is the prediction at x_j. The computer used for these numerical experiments was PIII 866 PC with 384MB RAM and Windows 2000 as the operating system.[14] We start with the simulated sinc data to study the role of β in our approach which is the main factor of advantage over the Huber's loss function and the quadratic loss function, and carry out the scaling results for SVR, BSVR and GPR; and then we employ the ARD Gaussian covariance function to carry out feature selection on robot arm data, and illustrate the predictive distribution on laser generated data; we also compare our method with GPR and SVR for generalization capability and computational cost on some benchmark data.

7.1 Sinc Data

The function $\text{sinc}(x) = |x|^{-1} \sin |x|$ is commonly used to illustrate SVR (Vapnik, 1995). Training and testing data sets were obtained by uniformly sampling data points from the interval $[-10, 10]$. Eight training data sets with sizes ranging from 50 to 4000 and a single common testing data set of 3000 cases were generated. The targets were corrupted by the noise generated by the noise model (14), using $C = 10$, $\epsilon = 0.1$ and $\beta = 0.3$.[15] From (16), the noise variance σ_n^2 is 0.026785 theoretically. The true noise variances σ_T^2 in each of the training data sets were computed and recorded in the second column of Table 1 as reference. The average squared noise in the testing data set is actually 0.026612, and the true value of average absolute noise is 0.12492.

We normalized the inputs of training data sets and keep the targets unchanged. We started from the default settings with a fixed value of $\beta = 0.3$. The training results were recorded in the

[13] κ_0 and κ_b are trivial for classical SVR in this case.

[14] The program *bisvm.exe* (version 4.2) and its source code we used for these numerical experiments can be accessed from http://guppy.mpe.nus.edu.sg/∼mpessk/papers/bisvm.zip.

[15] The simulated sinc data we generated can be accessed from http://guppy.mpe.nus.edu.sg/∼chuwei/data/sinc.zip. As for how to generate the noise distributed as the model (14), refer to Chu (2003).

upper part of Table 1. We find that the parameters C and ϵ approach the true value 10 and 0.1 respectively as the training sample size increases; σ_n^2, the variance of the additive noise that is estimated by (14) approaches σ_T^2 too; and the ASE on testing data set also approaches the true value of average squared noise. About 60% of training samples are selected as SVs. However, the training time increases heavily as the size of training data set becomes larger. The main reason is that the number of *off-bound* SVs that are involved in matrix inverse becomes larger. In the next experiment, we fixed β at a small value 0.1 and then carried out the training results, which were recorded in the lower part of Table 1. Comparing with the case of $\beta = 0.3$, we notice that the number of *off-bound* SVs decreases significantly for the case $\beta = 0.1$. That reduces the computational cost for the matrix inverse in the gradient evaluation for the evidence, and hence shortens the training time greatly. Moreover, the performance in testing does not worsen. Although β is not fixed at its true value, as the the size of training data increases, the estimated variance of the additive noise σ_n^2 still approaches σ_T^2 and the test ASE approaches to its true value too.

We also trained on the data set having 4000 examples, starting from the default settings with different β ranging from 0.001 to 1.0, and plotted the training results in Figure 3. Note that it is the Huber's loss function (11) when $\beta = 1.0$. We find that the number of off-bound SVs increases as β increases. The CPU time used to evaluate the evidence and its gradients increases significantly for β larger than 0.2, i.e., when the number of off-bound SVs greater than 1000. This makes the training on large-scale data sets very slow. The introduction of β makes it possible to reduce the number of off-bound SVs that involves in matrix inverse, and then saves lots of CPU time and memory. We also find that the evidence and test ASE is slightly unstable in the region of very small β, meanwhile the number of off-bound SVs becomes small. One reason might be that the change on the off-bound SVs set may cause fluctuation in evidence evaluation when the number of off-bound SVs is very few. Thus setting β at a very small value is not desirable. There exists a large range for the value of β (from 0.01 to 0.1) where the training speed is fast and the performance is good. The introduction of β makes it possible to reduce the number of off-bound SVs that involves in matrix inverse. This is one important advantage of our approach over the classical GPR in which the inverse of the full matrix is inevitable.

In the next experiments, we compared the generalization performance and the computational cost of GPR, SVR and BSVR on different size of the sinc simulated data. The size of training data set ranged from 10 to 1000. The targets were corrupted by additive Gaussian noise of variance 0.04, and 3000 noise-free samples were used as the test set for all the training data sets. At each size, we repeat the experiments 20 times to reduce the randomness in training data generation. The comparison of generalization performance is given in Figure 4(a)$\sim$Figure 4(f). BSVR and GPR yield better and more stable performance than SVR on small data sets. Clearly, when the number of training samples is small, Bayesian approaches are much better than SVR. GPR yields sightly better performance than BSVR when the size is less than 100, since GPR takes advantage on the Gaussian noise model and sparseness in BSVR may lose some information on small data sets. We presented the CPU time consumed by the three algorithms for the bunch of tasks, separately in Figure 4(g)$\sim$Figure 4(i). From the scaling results, we find that BSVR requires $\mathcal{O}(n^{2.36})$ computational cost, while GPR requires $\mathcal{O}(n^{3.05})$. This advantage of BSVR comes from the sparseness property in Bayesian inference that help us to tackle large data sets.

7.2 Robot Arm Data

The task in the robot arm problem is to learn the mapping from joint angles, x_1 and x_2, to the resulting arm position in rectangular coordinates, y_1 and y_2. The actual relationship between inputs and targets is as follows:

$$y_1 = 2.0 \cos x_1 + 1.3 \cos(x_1 + x_2) \quad \text{and} \quad y_2 = 2.0 \sin x_1 + 1.3 \sin(x_1 + x_2) \tag{46}$$

Targets are contaminated by independent Gaussian noise of standard deviation 0.05. The data set of robot arm problem we used here was generated by MacKay (1992) which contains 600 input-target pairs.[16] The first 200 samples in the data set were used as training set in all cases; the second 200 samples were used as testing set; the last 200 samples were not used. Two predictors were constructed for the two outputs separately in the training. We normalized the input data and keep the original target values, and then trained with ARD Gaussian model (42) starting from the default settings. The results are recorded in Table 2.

In the next experiment, four more input variables were added artificially (Neal, 1996), related to the inputs x_1 and x_2 in the original problem (46), x_3 and x_4 are copies of x_1 and x_2 corrupted by additive Gaussian noise of standard deviation 0.02, and x_5 and x_6 are irrelevant Gaussian noise inputs with zero mean, as follows: $x_1 = x_1$, $x_2 = x_2$, $x_3 = x_1 + 0.02 \cdot n_3$, $x_4 = x_2 + 0.02 \cdot n_4$, $x_5 = n_5$, $x_6 = n_6$, where n_3, n_4, n_5 and n_6 are independent Gaussian noise variables with zero mean and unit variance.[17] We normalized the input data and kept the original target values, and then trained an ARD Gaussian model (42) starting from the default settings. The results are recorded in Table 3. It is very interesting to look at the training results of the ARD parameters in the case of 6 inputs in Table 3. The values of the ARD parameters show nicely that the first two inputs are most important, followed by the corrupted inputs. The ARD parameters for the noise inputs shrink very fast in training. We also recorded the true variance of the additive Gaussian noise on y_1 and y_2 in the third column of Table 2 as reference, which are about 0.0025. Although the additive noise is Gaussian that is not consistent with our loss function in likelihood evaluation, we retrieve the noise variance properly. Meanwhile we keep sparseness in solution representation. About 50% $\sim$ 60% of the training samples are selected as SVs (refer to Table 2 and 3).

In Table 4, we compared the test ASE with that in other implementations, such as neural networks with Gaussian approximation by MacKay (1992) and neural networks with Monte Carlo by Neal (1996), and Gaussian processes for regression by Williams and Rasmussen (1996) etc. The expected test error of ASE based on knowledge of the true distribution is about 0.005. These results indicate that our approach gives a performance that is very similar to that given by well-respected techniques.[18]

[16] The robot arm data set generated by MacKay (1992) is available at http://wol.ra.phy.cam.ac.uk/mackay/bigback/dat/.

[17] The robot arm data set with six inputs we generated can be accessed from http://guppy.mpe.nus.edu.sg/~chuwei/data/robotarm.zip.

[18] Note that Monte Carlo methods sample hyperparameters hundreds of times according to $\mathcal{P}(\theta|\mathcal{D})$ and then average their individual predictions. Thus they have the advantage of reducing the uncertainty in hyperparameters. On the other hand, our approach takes the mode of $\mathcal{P}(\theta|\mathcal{D})$ as the optimal hyperparameters.

7.3 Laser Generated Data

SVR has been successfully applied to time series prediction (Müller et al., 1997). Here we choose the laser data to illustrate the error bar in predictions. The laser data has been used in the Santa Fe Time Series Prediction Analysis Competition.[19] A total of 1000 points of far-infrared laser fluctuations were used as the training data and 100 following points were used as testing data set. We normalized the training data set coordinate-wise, and used 8 consecutive points as the inputs to predict the next point. We chose Gaussian kernel (3) and started training from the default settings. β was fixed at 0.3. Figure 5 plots the predictions on testing data set and the error bars. Although the predictions of our model do not match the targets very well on the region (1051-1080), our model can reasonably provide larger error bars for these predictions. This feature is very useful in other learning fields, such as active learning.

7.4 Benchmark Comparisons

We compare our method BSVR with standard GPR (Williams, 1998) and classical SVR (Vapnik, 1995) upon generalization performance and computational cost on some benchmark data sets. The descriptions of these benchmark data sets we used are given as follows.

Boston Housing Data The "Boston Housing" data was collected in connection with a study of how air quality affects housing prices. The data concerns the median price in 1970 of owner-occupied houses in 506 census tracts within the Boston metropolitan area. Thirteen attributes pertaining to each census tract are available for use in prediction.[20] The objective is to predict the median house value. Following the method used by Tipping (2000) and Saunders et al. (1998),[21] the data set is partitioned into 481/25 training/testing splits randomly. This partitioning is carried out 100 times on the data. We cited the test ASE results reported by other methods in Table 5.

Computer Activity Data The computer activity data was collected from a Sun Sparcstation 20/712 with 128 Mbytes of memory running in a multi-user university department. The data set is composed of 8192 samples with 21 attributes.[22] The task is to predict the portion of time that CPUs run in user mode from all the 21 attributes. We partitioned the computer activity data into 2000/6192 training/testing splits randomly. The partitioning was repeated 10 times independently.

Abalone Data We normalize the abalone data[23] to zero mean and unit variance coordinate-wise, and then map the gender encoding (male/female/infant) into $\{(1,0,0),(0,1,0),(0,0,1)\}$. The normalized data set is split into 3000 training and 1177 testing data set randomly. The

[19]Full description can be found at URL: http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html.

[20]The original data can be found in StatLib, available at URL http://lib.stat.cmu.edu/datasets/boston.

[21]Saunders et al. (1998) used 80 cases in 481 training data as validation set to determine the kernel parameters.

[22]The data set and its full description can be accessed at http://www.cs.toronto.edu/~delve/data/comp-activ/.

[23]The data can be accessed via ftp://ftp.ics.uci.edu/pub/machine-learning-databases/abalone/.

partitioning is carried out 10 times independently. The objective is to predict the abalone's rings.

The results of BSVR using Gaussian covariance function against SVR are given in Table 6. SVR yields significantly better performance on the computer activity data than BSVR does.[24] The results of BSVR and GPR using ARD Gaussian covariance function are presented in Table 7. ARD feature selection greatly improves the generalization performance of BSVR on the computer activity data and the Boston housing data. BSVR performs significantly better than GPR in AAE on the Boston housing data and the abalone data. Meanwhile, BSVR is very efficient. Hence, BSVR with the benefit of sparseness can efficiently achieve very good generalization on reasonably large-scale data sets. If we could employ some scheme to cache part of the covariance matrix, the training time should be further reduced.

8 Conclusions

In this paper, we proposed a unifying loss function in a Bayesian design for support vector regression. The SILF is smooth and inherits most of the virtues of ϵ-ILF, such as insensitivity to outliers and sparseness in solution representation. In the Bayesian framework, we integrate support vector methods with Gaussian processes to keep the advantages of both. Various computational procedures are provided for the evaluation of MAP estimate and evidence of the hyperparameters. ARD feature selection and model adaptation are also implemented intrinsically in hyperparameter determination. Another benefit arising from the probabilistic formulation is the determination of error bars in making predictions. Furthermore, sparseness in the evidence evaluation and probabilistic prediction reduces the computational cost significantly and helps us to tackle reasonably large data sets. The results in numerical experiments show that the generalization ability is competitive with other well-respected techniques.

Acknowledgements

Wei Chu gratefully acknowledges the financial support provided by the National University of Singapore through Research Scholarship. A part of the revision work was done at Gatsby Computational Neuroscience Unit of University College London supported by the National Institutes of Health and its National Institute of General Medical Sciences division under Grant Number 1 P01 GM63208. The anonymous reviewers' thoughtful comments are gratefully appreciated.

References

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

[24]Note that GPR with Gaussian covariance function yields ASE 18.21 ± 1.07 and AAE 2.36 ± 0.046 on the computer activity data that is quite close to the test result of BSVR. SVR performs significantly better than BSVR and GPR on this dataset, possibly because the probabilistic models prefer relatively simple models when the noise level is quite high, while SVR selects the best matching model using cross validation.

Buntine, W. L. and A. S. Weigend. Bayesian back-propagation. *Complex Systems*, 5(6):603–643, 1991.

Byrd, R. H., P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995.

Chu, W. *Bayesian approach to support vector machines*. Ph.D. thesis, National University of Singapore, January 2003. http://guppy.mpe.nus.edu.sg/∼chuwei/paper/thesis.pdf.

Chu, W., S. S. Keerthi, and C. J. Ong. A unified loss function in Bayesian framework for support vector regression. In *Proceeding of the 18th International Conference on Machine Learning*, pages 51–58, 2001. http://guppy.mpe.nus.edu.sg/∼mpessk/svm/icml.pdf.

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

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.

Gao, J. B., S. R. Gunn, C. J. Harris, and M. Brown. A probabilistic framework for SVM regression and error bar estimation. *Machine Learning*, 46:71–89, March 2002.

Huber, P. J. *Robust Statistics*. John Wiley and Sons, New York, 1981.

Jebara, T. S. and T. S. Jaakkola. Feature selection and dualities in maximum entropy discrimination. In *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference (UAI-2000)*, pages 291–300, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

Keerthi, S. S. and S. K. Shevade. SMO algorithm for least squares SVM formulations. *Neural Computation*, 15(2):487–507, Feb. 2003.

Kimeldorf, G. S. and G. Wahba. Some results on Tchebycheffian spline function. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

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

Lampinen, J. and A. Vehtari. Bayesian approach for neural networks - reviews and case studies. *Neural Networks*, 14:257–274, 2001.

Law, M. H. and J. T. Kwok. Applying the Bayesian evidence framework to $\nu-$support vector regression. In *Proceeding of the Twelfth European Conference on Machine Learning*, pages 312–323, 2001a. Freiburg, Germany.

Law, M. H. and J. T. Kwok. Bayesian support vector regression. *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 239–244, 2001b. Key West, Florida, USA.

Luenberger, D. G. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, Menlo Park, California, 2nd edition, 1984.

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.

MacKay, D. J. C. Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3): 469–505, 1995.

Micchelli, C. A. Interpolation of scatter data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

Müller, K.-R., A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Using support vector machines for time series prediction. In Gerstner, W., A. Germond, M. Hasler, and J.-D. Nicoud, editors, *ICANN '97: Proc. of the Int. Conf. on Artificial Neural Networks*, pages 999–1004, 1997. Springer Berlin.

Neal, R. M. Bayesian training of backpropagation networks by the hybrid Monte Carlo method. Technical Report CRG-TR-92-1, Department of Statistics, University of Toronto, 1992.

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

Opper, M. and D. Saad. *Advanced Mean Field Methods - Theory and Practice*. The MIT Press, London, England, February 2001.

Pontil, M., S. Mukherjee, and F. Girosi. On the noise model of support vector regression. A.I. Memo 1651, Massachusetts Institute of Technology, Artificail Intelligence Laboratory, 1998.

Saunders, C., A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521, 1998.

Schölkopf, B., R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Proceedings of the Annual Conference on Computational Learning Theory*, 2001.

Schölkopf, B. and A. J. Smola. New support vector machines. NeuroCOLT2 Technical Report NC2-TR-1998-031, GMD FIRST, 1998.

Seeger, M. Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In *Advances in Neural Information Processing Systems*, volume 12, pages 603–609, 2000.

Shevade, S. K., S. S. Keerthi, C. Bhattacharyya, and K. R. K. Murthy. Improvements to the SMO algorithm for SVM regression. *IEEE Transactions on Neural Networks*, 11:1188–1194, Sept. 2000.

Smola, A. J. and B. Schölkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, GMD First, October 1998.

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

Tipping, M. E. The relevance vector machine. In Solla, S. A., T. K. Leen, and K.-R. Mller, editors, *Advances in Neural Information Processing Systems 12*, pages 652–658. MIT Press, 2000.

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.

Weston, J., S. Mukherjee, O. Chapelle, M. Pontil, and T. Poggio. Feature selection in SVMs. In Leen, T., T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, 2001. MIT Press.

Williams, C. K. I. Prediction with Gaussian processes: from linear regression to linear prediction and beyond. *Learning and Inference in Graphical Models*, 1998. Kluwer Academic Press.

Williams, C. K. I. and C. E. Rasmussen. Gaussian processes for regression. In Touretzky, D. S., M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 598–604, 1996. MIT Press.

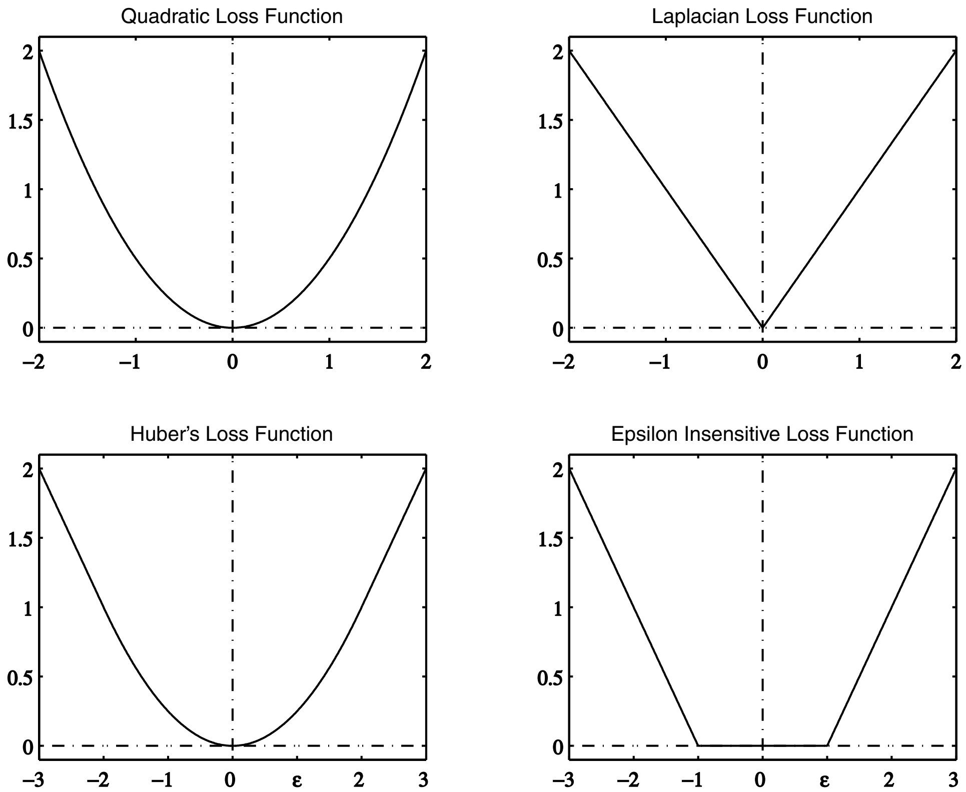

Figure 1: Graphs of popular loss functions, where ϵ is set at 1.

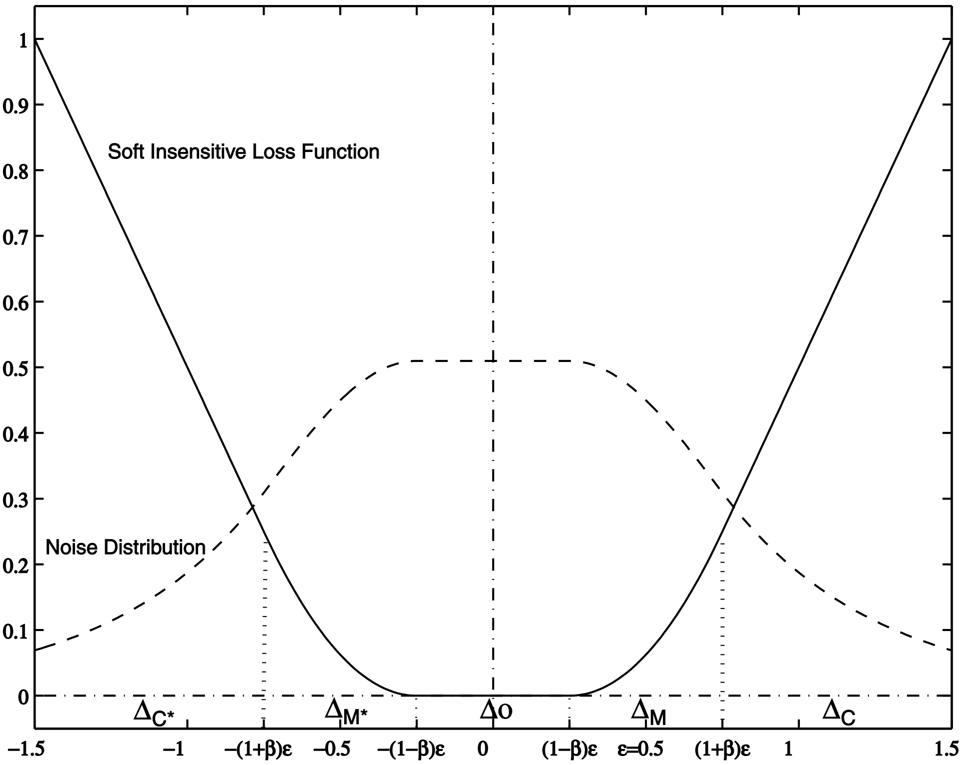

Figure 2: Graphs of soft insensitive loss function (solid curve) and its corresponding noise density function (dotted curve), where $\epsilon = 0.5$, $\beta = 0.5$ and $C = 2.0$ in the noise model.

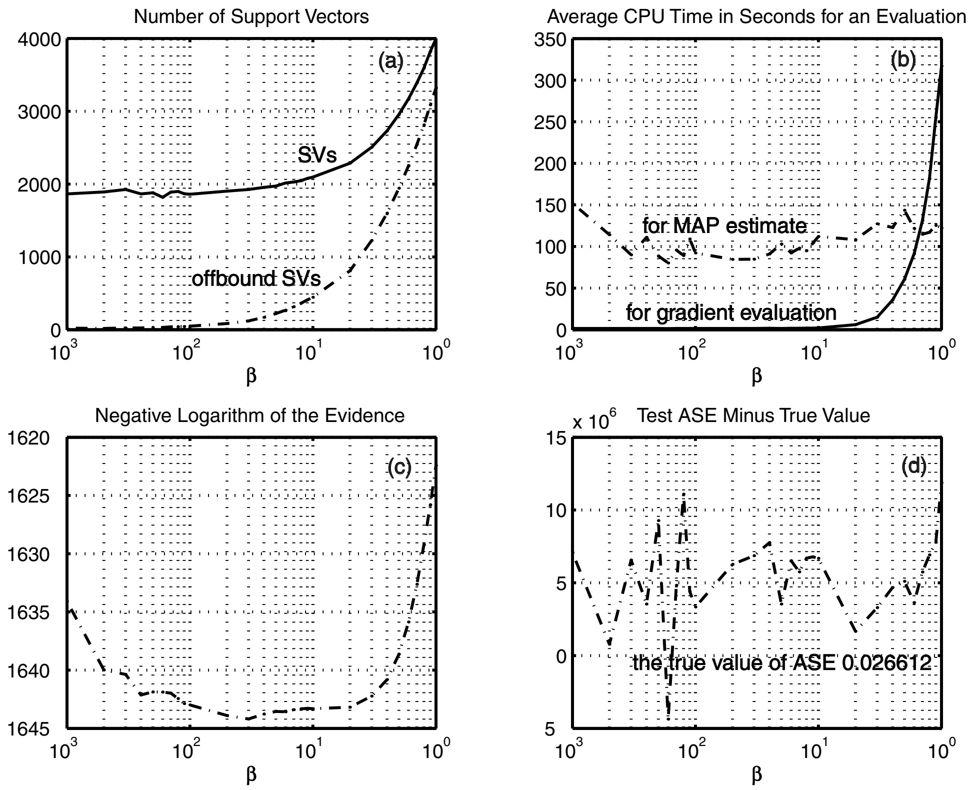

Figure 3: Graphs of training results with respect to different β for the 4000 sinc data set. The horizontal axis indicates the value of β in log-scale. The solid line in the graph (a) indicates the number of SVs, while the dotted line indicates the number of off-bound SVs. In the graph (b), the solid line indicate the CPU time in seconds used to evaluate evidence and its gradient, and the dotted line is the CPU time in seconds consumed for MAP estimate. In the graph (c), the dots indicate $-\ln \mathcal{P}(\mathcal{D}|\theta)$ in training results. In the graph (d), the dots indicate the average squared error (ASE) in testing minus the true value in the additive noise that is 0.026612.

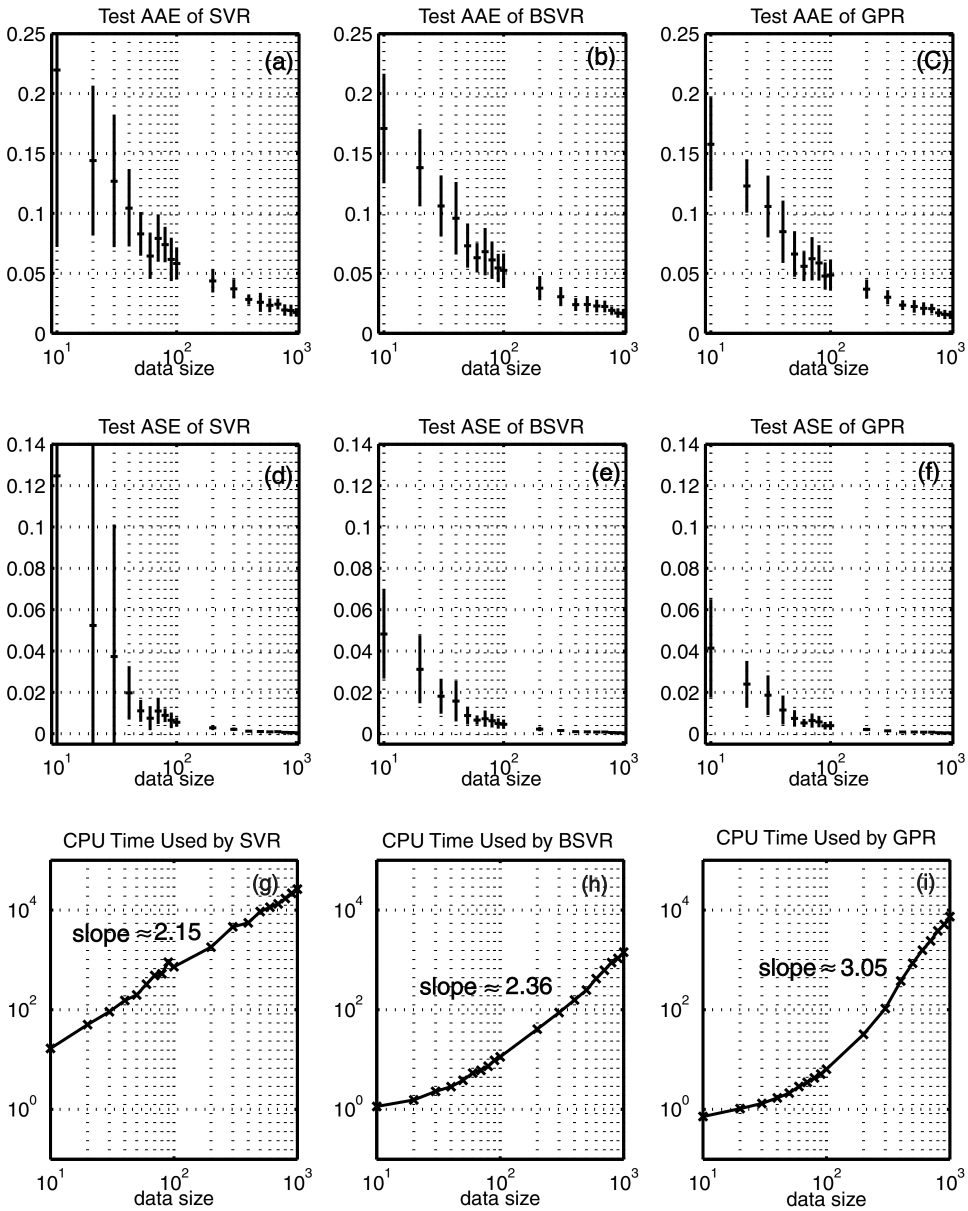

Figure 4: SVR, BSVR and GPR on the simulated sinc data at different training data size. The results of AAE and ASE are presented in the graph (a)∼(f) respectively. BSVR and GPR used evidence maximization to choose optimal hyperparameters, while five-fold cross validation was used for SVR. The position of cross denotes the average values over the 20 repetitions, and the vertical line indicates the standard deviation. In the graph (g)∼(i), we present the total CPU time in seconds consumed by these three approaches in training and test at different data sizes.

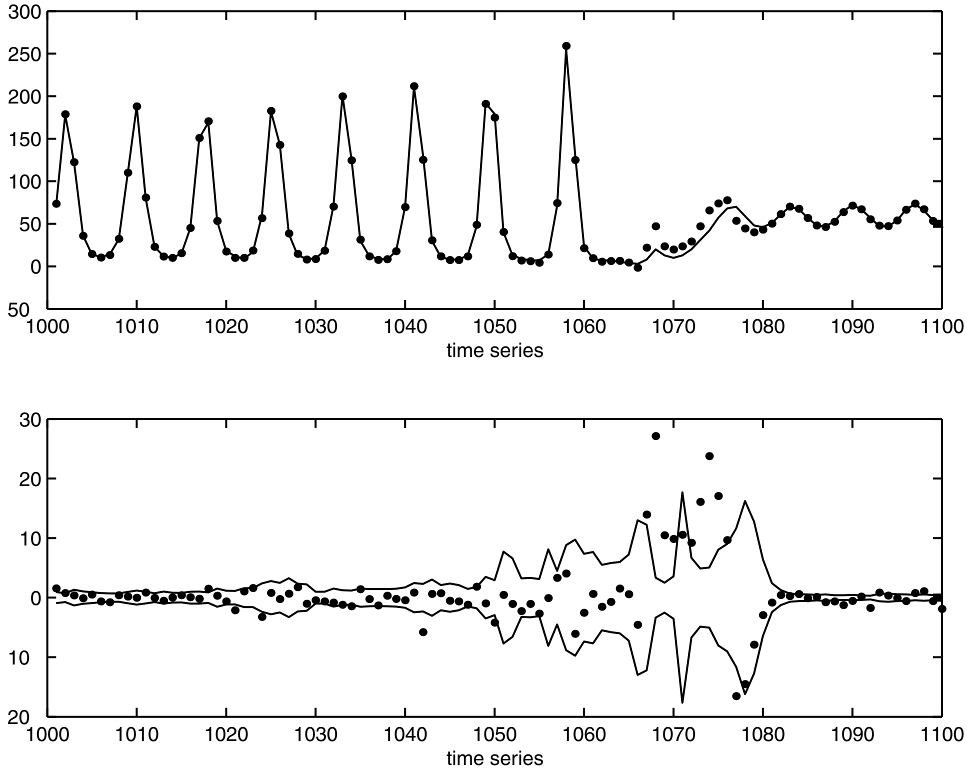

Figure 5: Graphs of our predictions on laser generated data. In the upper graph, the dots indicate our predictions on the testing data set and the solid curve describes the time series. In the lower graph, the dot indicates estimation error that is equal to prediction minus target, and solid curves indicate the error bars $\pm 2\sqrt{\sigma_t^2 + \sigma_n^2}$ in predictive distribution.

Table 1: Training results on sinc data sets with the fixed values, $\beta = 0.3$ or $\beta = 0.1$. σ_T^2 denotes the true value of noise variance in training data set; σ_n^2 denotes the noise variance in training data retrieved by (14); $-\ln \mathcal{P}(\mathcal{D}|\theta)$ denotes the negative log evidence of the hyperparameters as in (38); $\mathrm{SV_M}$ denotes the number of *off-bound* support vectors; $\mathrm{SV_C}$ denotes the number of *on-bound* support vectors; TIME denotes the CPU time in seconds consumed in the training; AAE is the average absolute error in test; ASE denotes the average squared error in test; the true value of average squared noise in the testing data set is 0.026612; the true value of average absolute noise in the testing data set is 0.12492.

β	Size	σ_T^2	C	ϵ	σ_n^2	κ	$-\ln \mathcal{P}_{\mathcal{D}\vert\theta}$	$\mathrm{SV_M}$	$\mathrm{SV_C}$	TIME	AAE	ASE
	50	.03012	15.95	.181	.02416	5.19	-1.3	23	4	0.15	.13754	.031194
	100	.03553	10.00	.136	.03152	5.85	-11.1	33	25	0.40	.13027	.028481
	300	.02269	11.16	.118	.02478	5.57	-113.7	90	87	5.95	.12642	.027189
0.3	500	.02669	9.36	.080	.02752	5.89	-174.8	135	218	12.9	.12544	.026765
	1000	.02578	9.90	.094	.02655	5.62	-389.9	270	388	63.0	.12537	.026834
	2000	.02639	10.01	.096	.02630	5.01	-808.8	539	768	436.2	.12509	.026661
	3000	.02777	9.96	.106	.02770	5.20	-1146.7	833	1052	1551.4	.12511	.026671
	4000	.02663	10.51	.111	.02609	5.76	-1642.2	1226	1280	3291.9	.12501	.026615
	50	.03012	6.70	.086	.05018	9.42	5.51	10	20	0.11	.13411	.030065
	100	.03553	12.07	.163	.02855	5.54	-10.1	19	25	0.53	.13366	.029728
	300	.02269	12.05	.124	.02300	5.92	-113.8	39	100	5.13	.12651	.027212
0.1	500	.02669	9.42	.080	.02715	5.78	-174.4	57	250	9.43	.12543	.026764
	1000	.02578	9.96	.095	.02631	6.09	-389.7	102	459	47.9	.12540	.026848
	2000	.02639	10.06	.096	.02600	5.06	-808.5	190	920	264.7	.12512	.026662
	3000	.02777	9.96	.108	.02774	5.34	-1142.6	287	1303	1070.4	.12509	.026673
	4000	.02663	10.41	.109	.02623	5.74	-1643.3	446	1650	2852.3	.12502	.026619

Table 2: Training results on the two-dimensional robot arm data set with the fixed value of $\beta = 0.3$. σ_T^2 denotes the true value of noise variance in the training data; σ_n^2 denotes the estimated value of the noise variance; $\mathrm{SV_M}$ denotes the number of *off-bound* support vectors; $\mathrm{SV_C}$ denotes the number of *on-bound* support vectors; TIME denotes the CPU time in seconds consumed in the training; AAE is the average absolute error in test; ASE denotes the average squared error in test.

y	$\sigma_T^{2\,(10^{-3})}$	C	ϵ	$\sigma_n^{2\,(10^{-3})}$	κ_1	κ_2	κ_b	$\mathrm{SV_M}$	$\mathrm{SV_C}$	TIME	AAE	$\mathrm{ASE}^{(10^{-3})}$
y_1	2.743	44.94	0.057	2.681	0.682	0.248	3.86	75	21	33.8	.03930	2.491
y_2	2.362	34.35	0.042	2.787	0.673	0.184	17.03	74	42	68.4	.04544	3.184

Table 3: Training results on the six-dimensional robot arm data set with the fixed value of $\beta = 0.3$. σ_n^2 denotes the estimated value of the noise variance; $\mathrm{SV_M}$ denotes the number of *off-bound* support vectors; $\mathrm{SV_C}$ denotes the number of *on-bound* support vectors; AAE is the average absolute error in test; ASE denotes the average squared error in test.

y	$\sigma_n^2{}^{(10^{-3})}$	κ_1	κ_2	$\kappa_3{}^{(10^{-2})}$	$\kappa_4{}^{(10^{-2})}$	$\kappa_5{}^{(10^{-5})}$	$\kappa_6{}^{(10^{-5})}$	κ_b	$\mathrm{SV_M}$	$\mathrm{SV_C}$	AAE	$\mathrm{ASE}^{(10^{-3})}$
y_1	2.696	.667	.248	.287	.0087	0.01	0.01	2.36	74	27	.03907	2.477
y_2	2.779	.603	.222	8.41	.904	0.01	0.01	23.53	60	77	.04622	3.160

Table 4: Comparison with other implementation methods on testing ASE of the robot arm positions. INPUTS denotes the number of inputs. ASE denotes the average squared error in testing.

IMPLEMENTATION METHOD	INPUTS	$\mathrm{ASE}^{(10^{-3})}$
Gaussian Approximation of MacKay		
Solution with highest evidence	2	5.73
Solution with lowest test error	2	5.57
Hybrid Monte Carlo of Neal	2	5.47
	6	5.49
Gaussian Processes of Williams and Rasmussen	2	5.63
	6	5.69
GPR using Evidence Maximization		
with Gaussian Covariance Function	2	5.83
with ARD Gaussian	2	5.70
with ARD Gaussian	6	5.70
SVR using Gaussian Covariance Function		
$\epsilon = 0.1$	2	7.46
$\epsilon = 0.05$	2	6.82
$\epsilon = 0.01$	2	5.84
BSVR with $\beta = 0.3$		
Gaussian Covariance Function	2	5.89
ARD Gaussian	2	5.68
ARD Gaussian	6	5.64

Table 5: Comparison with Ridge Regression (Saunders et al., 1998), Relevance Vector Machine Tipping (2000), GPR and SVR on price prediction of the Boston Housing data set. ASE denotes the average squared test error.

IMPLEMENTATION METHOD	KERNEL TYPE	ASE
Ridge Regression	Polynomial	10.44
Ridge Regression	Splines	8.51
Ridge Regression	ANOVA Splines	7.69
Relevance Vector Machine	Gaussian	7.46
SVR	Gaussian	10.27
GPR	Gaussian	9.13
BSVR with $\beta = 0.3$	Gaussian	12.34
GPR	ARD Gaussian	8.32
BSVR with $\beta = 0.3$	ARD Gaussian	6.99

Table 6: Training results of BSVR and standard SVR on the benchmark data sets. Both of them used the Gaussian covariance function (3). TIME denotes the total CPU time in hours consumed by BSVR for all partitions of that data set, and SVR is the corresponding value of SVR. ASE denotes the test ASE of BSVR averaged over all partitions of that data set together with the standard deviation, and SVR-ASE is the corresponding element of SVR. AAE denotes the test AAE of BSVR, and SVR-AAE is the corresponding element of SVR. The p-value is for the paired $t-$test on test error. 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	SVR	TIME	SVR-ASE	ASE	p-value	SVR-AAE	AAE	p-value
Housing	22.0	0.9	10.27±7.21	12.34±9.20	0.078	2.13±0.48	2.19±0.48	0.32
Computer	45.6	4.7	**13.80±0.93**	17.59±0.98	5.5×10^{-8}	2.28±0.04	2.33±0.05	0.026
Abalone	67.8	6.5	0.441±0.021	0.438±0.024	0.78	0.455±0.0088	0.454±0.0086	0.95

Table 7: Training results of BSVR and standard GPR on the benchmark data sets. Both of them used the ARD Gaussian covariance function (42). TIME denotes the total CPU time in hours consumed by BSVR for training on all partitions of that data set, and GPR is the corresponding value of GPR. ASE denotes the test ASE of BSVR averaged over all partitions of that data set together with the standard deviation, and GPR-ASE is the corresponding element of GPR. AAE denotes the test AAE of BSVR, and GPR-AAE is the corresponding element of GPR. The p-value is for the paired $t-$test on test error. 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	GPR	TIME	GPR-ASE	ASE	p-value	GPR-AAE	AAE	p-value
Housing	2.4	2.2	8.32±4.35	6.99±4.38	0.032	2.01±0.40	**1.86±0.37**	0.0060
Computer	23.0	12.1	5.58±0.25	5.80±0.27	0.070	1.686±0.023	1.687±0.026	0.99
Abalone	43.6	13.6	0.428±0.022	0.432±0.023	0.73	0.463±0.0087	**0.451±0.0095**	0.0094