BAYESIAN APPROACH TO SUPPORT VECTOR MACHINES

CHU WEI

(Master of Engineering)

A DISSERTATION SUBMITTED FOR

THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2003

To my grandmother

Acknowledgements

I wish to express my deepest gratitude and appreciation to my two supervisors, Prof. S. Sathiya Keerthi and Prof. Chong Jin Ong for their instructive guidance and constant personal encouragement during every stage of this research. I greatly respect their inspiration, unwavering examples of hard work, professional dedication and scientific ethos.

I gratefully acknowledge the financial support provided by the National University of Singapore through Research Scholarship that makes it possible for me to study for academic purpose.

My gratitude also goes to Mr. Yee Choon Seng, Mrs. Ooi, Ms. Tshin and Mr. Zhang for the helps on facility support in the laboratory so that the project can be completed smoothly.

I would like to thank my family for their continued love and support through my student life.

I am also fortunate to meet so many talented fellows in the Control Laboratory, who make the three years exciting and the experience worthwhile. I am sincerely grateful for the friendship and companion from Duan Kaibo, Chen Yinghe, Yu weimiao, Wang Yong, Siah Keng Boon, Zhang Zhenhua, Zheng Xiaoqing, Zuo Jing, Shu Peng, Zhang Han, Chen Xiaoming, Chua Kian Ti, Balasubramanian Ravi, Zhang Lihua, for the stimulating discussion with Shevade Shirish Krishnaji, Rakesh Menon, and Lim Boon Leong. Special thanks to Qian Lin who makes me happier in the last year.

Table of contents

Summary

In this thesis, we develop Bayesian support vector machines for regression and classification. Due to the duality between reproducing kernel Hilbert space and stochastic processes, support vector machines can be integrated with stationary Gaussian processes in a probabilistic framework. We propose novel loss functions with the purpose of integrating Bayesian inference with support vector machines smoothly while preserving their individual merits, and then in this framework we apply popular Bayesian techniques to carry out model selection for support vector machines. The contributions of this work are two-fold: for classical support vector machines, we follow the standard Bayesian approach using the new loss function to implement model selection, by which it is convenient to tune a large number of hyperparameters automatically; for standard Gaussian processes, we introduce sparseness into Bayesian computation through the new loss function which helps to reduce the computational burden and hence makes it possible to tackle large-scale data sets.

For regression problems, we propose a novel loss function, namely soft insensitive loss function, which is a unified non-quadratic loss function with the desirable characteristic of differentiability. We describe a Bayesian framework in stationary Gaussian processes together with the soft insensitive loss function in likelihood evaluation. Under this framework, the maximum a posteriori estimate on the function values corresponds to the solution of an extended support vector regression problem. Bayesian methods are used to implement model adaptation, while keeping the merits of support vector regression, such as quadratic programming and sparseness. Moreover, we put forward error bar in making predictions. Experimental results on simulated and real-world data sets indicate that the approach works well. Another merit of the Bayesian approach is that it provides a feasible solution to large-scale regression problems.

For classification problems, we propose a novel differentiable loss function called trigonometric loss function with 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 classifiers, such as sparseness and convex programming. Moreover, we put forward class probability in making predictions. Experimental results on benchmark data sets indicate the usefulness of this approach.

In this thesis, we focus on regression problems in the first four chapters, and then extend our discussion to binary classification problems. The thesis is organized as follows: in Chapter 1 we review the current techniques for regression problems, and then clarify our motivations and intentions; in Chapter 2 we review the popular loss functions, and then propose a new loss function, soft insensitive loss function, as a unified loss function and describe some of its useful properties; in Chapter 3 we review Bayesian designs on generalized linear models that include Bayesian neural networks and Gaussian processes; a detailed Bayesian design for support vector regression is discussed in Chapter 4; we put forward a Bayesian design for binary classification problems in Chapter 5 and we conclude the thesis in Chapter 6.

Nomenclature

A^T	transposed matrix (or vector)
A^{-1}	inverse matrix
A_{ij}	the ij-th entry of the matrix A
C	a parameter in likelihood, $C > 0$
$Cov[\cdot, \cdot]$	covariance of two random variables
$E[\cdot]$	expectation of a random variable
$K(\cdot, \cdot)$	kernel function
$Var[\cdot]$	variance of a random variable
$\boldsymbol{\alpha}, \boldsymbol{\alpha}^*$	column vectors of Lagrangian multipliers
α_i, α_i^*	Lagrangian multipliers
δ	the noise
δ_{ij}	the Kronecker delta
det	the determinant of the matrix
$\ell(\cdot)$	loss function
$\boldsymbol{\eta}$	the column vector of Lagrangian multipliers
η_i	Lagrangian multiplier
λ	regularization parameter, $\lambda > 0$
$\mathbb{R}$	the set of reals
$\mathbf{H}$	Hessian matrix
$\mathbf{I}$	Identity matrix
$\mathbf{w}$	weight vector in column
$\mathcal{D}$	the set of training samples, $\{(x_i, y_i) \mid i = 1, \ldots, n\}$
$\mathcal{F}(\cdot)$	distribution function
$\mathcal{L}$	the likelihood
$\mathcal{N}(\mu, \sigma^2)$	normal distribution with mean μ and variance σ^2
$\mathcal{P}(\cdot)$	probability density function or probability of a set of events
$\lvert \cdot \rvert$	the determinant of the matrix
$\nabla_{\mathbf{w}}$	differential operator with respect to $\mathbf{w}$

$\|\cdot\|$	the Euclidean norm
θ	model parameter (hyperparameter) vector
$\boldsymbol{\xi}, \boldsymbol{\xi}^*$	vectors of slack variables
ξ_i, ξ_i^*	slack variables
b	bias or constant offset, $b \in \mathbb{R}$
d	the dimension of input space
i, j	indices
n	the number of training samples
x	an input vector, $x \in \mathbb{R}^d$
x^ι	the ι-th dimension (or feature) of x
$\boldsymbol{x}$	the set of input vectors, $\{x_1, x_2, \ldots, x_n\}$
y	target value $y \in \mathbb{R}$, or class label $y \in \{\pm 1\}$
$\boldsymbol{y}$	target vector in column or the set of target, $\{y_1, y_2, \ldots, y_n\}$
(x, y)	a pattern
$f_x, f(x)$	the latent function at the input x
$\boldsymbol{f}$	the column vector of latent functions
$z \in (a, b)$	interval $a < z < b$
$z \in (a, b]$	interval $a < z \leq b$
$z \in [a, b]$	interval $a \leq z \leq b$
i	imaginary unit

List of Figures

List of Tables

Chapter 1

Introduction and Review

There are many problems in science, statistics and technology which can be effectively modelled as the learning of an input-output mapping given some data set. The mapping usually takes the form of some unknown function between two spaces as $f : \mathbb{R}^d \to \mathbb{R}$. The given data set $\mathcal{D}$ is composed of n independent, identically distributed (i.i.d.) samples, i.e., the pairs (x_1, y_1), $\ldots$, (x_n, y_n) which are obtained by randomly sampling the function f in the input-output space $\mathbb{R}^d \times \mathbb{R}$ according to some unknown joint probability density function $\mathcal{P}(x, y)$. In the presence of additive noise, the generic model for these pairs can be written as

$$y_i = f(x_i) + \delta_i, \qquad i = 1, \ldots, n, \tag{1.1}$$

where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, and the δ_i are i.i.d. random variables, whose distributions are usually unknown. *Regression* aims to infer the underlying function f, or an estimate of it from the finite data set $\mathcal{D}$.[1]

In trivial cases, the parametric model of the underlying function is known, and is therefore determined uniquely by the value of a parameter vector θ. We denote the parametric form as $f(x; \theta)$, and assume that the additive noise δ in measurement (1.1) has some known distribution with probability density function $\mathcal{P}(\delta)$, which is usually Gaussian. Due to the dependency of $\mathcal{P}(\delta)$ on θ, we explicitly rewrite $\mathcal{P}(\delta)$ as $\mathcal{P}(\delta; \theta)$ or $\mathcal{P}(y - f(x; \theta))$.[2] Our problem becomes the use of the information provided by the samples to obtain good estimates of the unknown parameter vector θ in the parametric model $f(x; \theta)$. For regular models, the method of *maximum likelihood*

[1] Classification or pattern recognition could be regarded as a special case of regression problems in which the targets y take only limited values, usually binary values $\{-1, +1\}$. We will discuss the case later in a separate chapter.

[2] We prefer to write $\mathcal{P}(\delta; \theta)$ here, since the θ is treated as ordinary parameters for maximum likelihood analysis. The notation $\mathcal{P}(\delta|\theta)$ implies that θ is random variables, which is suitable for Bayesian analysis.

could be used to find the values $\hat{\theta}$ of the parameters which is "most likely" to have produced the samples. Suppose that the data $\mathcal{D} = \{(x_i, y_i) \mid i = 1, \ldots, n\}$ have been independently drawn. The probability of observing target values $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$ at corresponding input points $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ can be stated as a conditional probability $\mathcal{P}(\boldsymbol{y}|\boldsymbol{x}; \theta)$. Here each pair (x_i, y_i) is associated with a noise value δ_i. For any θ, the probability of observing these discrete points $\mathcal{D}$ can be given as

$$\mathcal{P}(\mathcal{D}; \theta) = \mathcal{P}(\boldsymbol{y}|\boldsymbol{x}; \theta)\mathcal{P}(\boldsymbol{x}). \tag{1.2}$$

where $\mathcal{P}(\boldsymbol{y}|\boldsymbol{x}; \theta) = \prod_{i=1}^{n} \mathcal{P}(\delta = \delta_i; \theta) = \prod_{i=1}^{n} \mathcal{P}(y_i - f(x_i; \theta))$ and $\mathcal{P}(\boldsymbol{x})$ is independent of θ. Viewed as a function of θ, $\mathcal{P}(\boldsymbol{y}|\boldsymbol{x}; \theta)$ is called the likelihood of θ with respect to the set of samples. The maximum likelihood estimate of θ is, by definition, the value $\hat{\theta}$ that maximizes $\mathcal{P}(\boldsymbol{y}|\boldsymbol{x}; \theta)$. Thus the value $\hat{\theta}$ can be obtained from the set of equations $\frac{\partial \mathcal{P}(\boldsymbol{y}|\boldsymbol{x}; \theta)}{\partial \theta}|_{\theta=\hat{\theta}} = 0$.

1.1 Generalized Linear Models

In general, the model of the underlying function is unknown. We have to develop a *regression function* from some universal approximator, which has sufficient capacity to arbitrarily closely approximate any continuous input-output mapping function defined on a compact domain. The universal approximation theorem (Park and Sandberg, 1991; Cybenko, 1989) states that both multilayer perceptrons (MLP) with single hidden layer and radial basis function (RBF) networks are universal approximators.

1. MLP: the regression function given by a MLP network with single hidden layer can be written as

$$f(x; \Theta) = \sum_{i=1}^{m} w_i \varphi(x; \nu_i) + b \tag{1.3}$$

   where m is the number of hidden neurons, $\varphi(x; \nu_i)$ is the activation function, and ν_i is the weight vector in the i-th hidden neuron. Θ denotes the set of all parameters that include hidden-to-output weights $\{w_i\}_{i=1}^{m}$, input-to-hidden weights $\{\nu_i\}_{i=1}^{m}$ and the bias b. Logistic function $\varphi(x; \nu) = \frac{1}{1+\exp(-\nu \cdot x)}$ or hyperbolic tangent function $\varphi(x; \nu) = \tanh(\nu \cdot x)$ is commonly used as the activation function in the hidden neurons.

2. RBF: the regression function given by a RBF network can be written as

$$f(x; \Theta) = \sum_{i=1}^{m} w_i \varphi(x; \mu_i, \sigma_i) + b \tag{1.4}$$

where $\varphi(x;\mu_i,\sigma_i)$ is a radial basis function, μ_i is the center of the radial basis function and σ_i denotes other parameters in the function. Green's functions (Courant and Hilbert, 1953), especially the Gaussian function $G(\|x-\mu\|;\sigma) = \exp\left(-\frac{\|x-\mu\|^2}{2\sigma^2}\right)$ which is translationally and rotationally invariant, are widely used as radial basis functions. In the Gaussian function, the parameter σ is usually called the spread (Haykin, 1999). As a particular case, the number of hidden neurons is chosen same as the size of training data, i.e. $m = n$, and the centers are fixed at $\mu_i = x_i \ \forall i$, which is known as generalized RBF (Poggio and Girosi, 1990).

The regression function of the two networks could be generalized into a unified formulation as

$$f(x;\Theta) = \sum_{i=1}^{m} w_i \varphi(x;\theta_i) + b = \sum_{i=0}^{m} w_i \varphi(x;\theta_i) \tag{1.5}$$

where $w_0 = b$, $\varphi(x;\theta_i)$ is a set of basis functions with $\varphi(x;\theta_0) = 1$, and Θ denotes the set of free parameters in the model. The regression function in (1.5) is a parameterized linear superposition of basis functions, which is usually referred to as *generalized linear models*. We give a architectural graph of generalized linear models in Figure 1.1. Depending on the choice of the basis function, different networks, such as MLP with single hidden layer and RBF, could be obtained.

In order to choose the best available regression function for the given data, we usually measure the discrepancy between the target y to a given input x and the response $f(x;\Theta)$ by some loss function $\ell(y, f(x;\Theta))$,[3] and consider the expected value of the loss, given by the risk functional

$$\mathcal{R}(\Theta) = \int\int \ell(y, f(x;\Theta))\mathcal{P}(x,y)\,dx\,dy. \tag{1.6}$$

Since only finite samples are available, the risk functional $\mathcal{R}(\Theta)$ is usually replaced by the so-called empirical risk functional

$$\mathcal{R}_{emp}(\Theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i;\Theta)). \tag{1.7}$$

The solution to the minimization of $\mathcal{R}_{emp}(\Theta)$ could be selected as the desired regression function. This principle of minimizing the empirical risk functional (1.7) on the basis of empirical data is referred to as the *empirical risk minimization* (ERM) inductive principle.

[3] Usually, there is a close relationship between the loss function $\ell(y|f(x;\Theta))$ and the likelihood function $\mathcal{P}(y|f(x;\Theta))$, which is $\mathcal{P}(y|f(x;\Theta)) \propto \exp(-C \cdot \ell(y, f(x;\Theta)))$ where C is a parameter greater than zero.

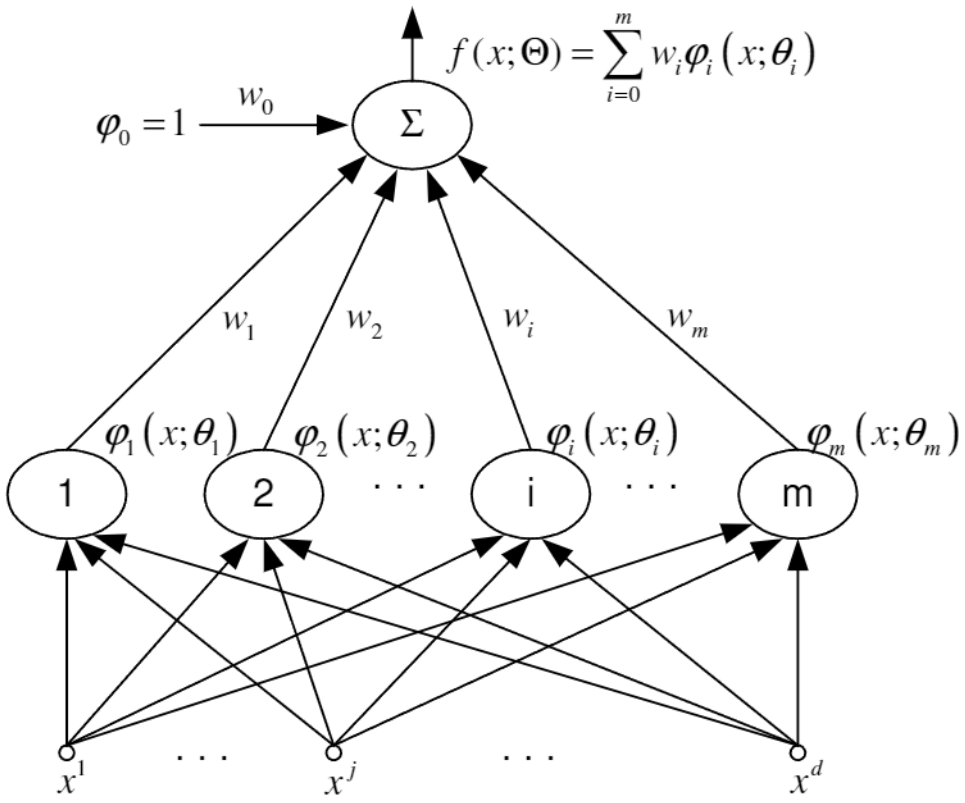

Figure 1.1: A architectural graph for generalized linear models.

However, for a universal approximator that has sufficient power to represent any arbitrary continuous function, the minimization problem in ERM is obviously ill-posed, because it will yield an infinite number of solutions that give a zero value for $\mathcal{R}_{emp}(\Theta)$. A more complex model that is of more powerful representational capacity typically fits the empirical data better. Preferring these "best fit" models leads us to choose implausibly over-parameterized models, which might provide poor prediction for future data.

1.2 Occam's Razor

There is a general philosophical principle known as Occam's razor for model selection, which is highly influential when applied to various scientific theories.

> *No more things should be presumed to exist than are absolutely necessary.*
>
> —"Occam's razor" principle attributed to W. Occam (c. 1285−1349).

In the light of the Occam's razor, unnecessary complex models should not be preferred to simpler ones. This intuitive principle could be applied quantitatively in several ways.

1.2.1 Regularization

Regularization theory, which was proposed by Tikhonov (1963) to solve the ill-posed problems,[4] could incarnate the Occam's razor principle as an optimization problem on the tradeoff between model complexity and the empirical risk. This optimal trade-off is realized by minimizing the regularized functional

$$\mathcal{R}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i;\Theta)) + \lambda \cdot \Phi(f) \tag{1.8}$$

The L_2 loss function $\|y - f(x;\Theta)\|^2$ is widely used for the empirical risk in the first data-dependent term; $\Phi(f)$ is a stabilizer with certain properties (Tikhonov and Arsenin, 1977; Girosi et al., 1995) to measure the model complexity; and the positive constant λ is a regularization parameter, which represents the relative importance of the model complexity with respect to the performance measure.

Regression problem in classical regularization theory (Evgeniou et al., 1999) could be formulated as a variational problem of finding the function f in a reproducing kernel Hilbert space (RKHS)[5] that minimizes the regularized functional

$$\mathcal{R}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i;\Theta)) + \lambda \cdot \|f\|^2_{\mathrm{RKHS}} \tag{1.9}$$

where $\|f\|^2_{\mathrm{RKHS}}$ is a norm in the RKHS. The RKHS is defined by the positive-definite function $K(x_i, x_j)$, namely reproducing kernel, on $\mathbb{R}^d \times \mathbb{R}^d$.[6]

Mercer-Hilbert-Schmidt theorem (Wahba, 1990; Riesz and Sz.-Nagy, 1955) says that there exists an orthonormal sequence of continuous eigenfunctions, $\phi_1, \phi_2, \ldots$ on $\mathbb{R}^d$ and eigenvalues $\upsilon_1 \geq \upsilon_2 \geq \ldots \geq 0$, with

$$\int_{\mathbb{R}^d} K(x_i, x_j)\phi_\tau(x_j)\, dx_j = \upsilon_\tau \phi_\tau(x_i), \tau = 1, 2, \ldots,$$

$$K(x_i, x_j) = \sum_{\tau=1}^{\infty} \upsilon_\tau \phi_\tau(x_i)\phi_\tau(x_j), \tag{1.10}$$

$$\int_{\mathbb{R}^d}\int_{\mathbb{R}^d} K^2(x_i, x_j)\, dx_i\, dx_j = \sum_{\tau=1}^{\infty} \upsilon_\tau^2 < \infty,$$

under the condition that the positive-definite function $K(x_i, x_j)$ is continuous and

[4] The ill-posed problem is of the type $\mathcal{A}f = D$, where $\mathcal{A}$ is an (linear) operator, f is the desired solution in a metric space E_1, and D is the 'data' in a metric space E_2. Even if there exists a unique solution to this equation, a small deviation on the right-hand side of this equation can cause large deviations in the solutions.

[5] For a modern account on the theory of RKHS, please refer to Saitoh (1988) or Small and McLeish (1994).

[6] A kernel function $K(x_i, x_j)$ can be any symmetric function satisfying Mercer's condition (Courant and Hilbert, 1953)

$$\int_{\mathbb{R}^d}\int_{\mathbb{R}^d} K^2(x_i, x_j)\, dx_i\, dx_j < \infty.$$

In the RKHS, the function f possesses a unique representation (Wahba, 1990)

$$f(x) = \sum_{\tau=1}^{\infty} \gamma_\tau \phi_\tau(x), \tag{1.11}$$

where $\gamma_\tau = \int_{\mathbb{R}^d} f(t)\phi_\tau(t)\, dt$, and its norm $\|f\|^2_{\text{RKHS}} = \sum_{\tau=1}^{\infty} \frac{\gamma_\tau^2}{\upsilon_\tau}$. Thus, the functional $\mathcal{R}(f)$ in (1.9) could be regarded as a function of the coefficients γ_τ. Suppose that the partial derivative of the loss function ℓ in (1.9) with respect to γ_τ exists. In order to minimize $\mathcal{R}(f)$ we take its derivative with respect to γ_τ and set it equal to zero, obtaining the following

$$\frac{1}{n}\sum_{i=1}^{n} \frac{\partial \ell(y_i, f(x_i;\Theta))}{\partial f(x_i;\Theta)} \cdot \phi_\tau(x_i) + 2\lambda \cdot \frac{\gamma_\tau}{\upsilon_\tau} = 0.$$

where $\frac{\partial \ell(y_i, f(x_i;\Theta))}{\partial f(x_i;\Theta)}$ denotes the partial derivative of the loss function with respect to $f(x_i;\Theta)$. Let us define the following set of unknowns: $w_i = -\frac{1}{2n\lambda} \cdot \frac{\partial \ell(y_i, f(x_i;\Theta))}{\partial f(x_i;\Theta)}$. Then the coefficients γ_τ could be expressed as a function of the w_i:

$$\gamma_\tau = \upsilon_\tau \sum_{i=1}^{n} w_i \cdot \phi_\tau(x_i).$$

Together with (1.10) and (1.11), the solution of the variational problem (1.9), therefore, has the dual form:

$$f(x) = \sum_{\tau=1}^{\infty} \gamma_\tau \phi_\tau(x) = \sum_{\tau=1}^{\infty} \upsilon_\tau \sum_{i=1}^{n} w_i \cdot \phi_\tau(x_i) \cdot \phi_\tau(x) = \sum_{i=1}^{n} w_i K(x_i, x).$$

This shows that, for any differentiable loss function, the solution of the regularization functional $\mathcal{R}(f)$, is always a linear superposition of kernel functions, one for each data point. This elegant form of a minimizer of (1.9) 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 as any strictly monotonically increasing function $\ell : \mathbb{R} \to [0, +\infty)$.

The choice of the optimal regularization parameter λ and other free parameters in kernel functions is an important issue in regularization techniques and typically cross validation (Kearns et al., 1995; Wahba, 1990) or other heuristic schemes are used for that. We will further discuss this issue in Chapter 4 and Chapter 5.

1.2.2 Bayesian Learning

Bayesian methods could also quantify Occam's razor automatically (Gull, 1988; MacKay, 1992c). The fundamental concept of Bayesian analysis is that the plausibility of alternative hypotheses is represented by probability, and inference is performed by evaluating those probabilities. Suppose that we have collected a set of hypothesis classes, $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_L$, that compete to account for the data we are given. Our initial knowledge about the plausibility of these models is quantified by a list of prior probability, $\mathcal{P}(\mathcal{M}_1)$, $\mathcal{P}(\mathcal{M}_2)$, ..., $\mathcal{P}(\mathcal{M}_L)$, which sum to 1. Each model $\mathcal{M}_i$ makes predictions about how probable different data sets are, if this model is the true. The accuracy of the model's predictions is evaluated by a conditional probability $\mathcal{P}(\mathcal{D}|\mathcal{M}_i)$, the probability of $\mathcal{D}$ given $\mathcal{M}_i$. When we observe the actual data $\mathcal{D}$, *Bayes' theory* describes how we should update our beliefs of the model on the basis of the data $\mathcal{D}$. Bayes' theory can be written as

$$\mathcal{P}(\mathcal{M}_i|\mathcal{D}) = \frac{\mathcal{P}(\mathcal{D}|\mathcal{M}_i)\mathcal{P}(\mathcal{M}_i)}{\mathcal{P}(\mathcal{D})} \tag{1.12}$$

where $\mathcal{P}(\mathcal{M}_i|\mathcal{D})$, the posterior probability, represents our final beliefs about the model $\mathcal{M}_i$ given that we have observed $\mathcal{D}$; the denominator $\mathcal{P}(\mathcal{D})$ is a normalizing constant which makes $\mathcal{P}(\mathcal{M}_i|\mathcal{D})$ of all the models add up to 1, and the data-dependent term $\mathcal{P}(\mathcal{D}|\mathcal{M}_i)$ is the evidence for the model $\mathcal{M}_i$. Notice that the subjective prior probability $\mathcal{P}(\mathcal{M}_i)$ expresses how plausible we thought the alternative models were before the observational data arrived. Since we usually have no reason to assign strongly different $\mathcal{P}(\mathcal{M}_i)$, the posterior probability $\mathcal{P}(\mathcal{M}_i|\mathcal{D})$ will typically be overwhelmed by the objective term, the evidence. Thus, in these cases the evidence could be used as a criterion to assign a preference to the alternative models $\mathcal{M}_i$.

As the quantity for comparing alternative models for Bayesians, the evidence naturally embodies Occam's razor that has been elucidated by MacKay (1992c). Let us use MLP networks with a single hidden layer (1.3) as the model (hypothesis class) to account for the training data $\mathcal{D}$. A MLP network with m hidden neurons is denoted as $\mathcal{M}_m$. The model set is composed by MLP networks with different hidden neurons $\{\mathcal{M}_m\}$. Each model $\mathcal{M}_m$ is defined by a set of the weight vector $\mathbf{w}$, which includes hidden-to-output weights $\{w_i\}_{i=1}^m$, input-to-hidden weights $\{\nu_i\}_{i=1}^m$ and the bias b as in (1.3), and associated two probability distributions: a prior distribution $\mathcal{P}(\mathbf{w}|\mathcal{M}_m)$ which expresses what values the model's weights might plausibly take; and the model's descriptions to the data set $\mathcal{D}$ when its weights have been specified a particular value $\mathbf{w}$, $\mathcal{P}(\mathcal{D}|\mathbf{w}, \mathcal{M}_m)$. The evidence of the model $\mathcal{M}_m$ can be obtained by an integral over all weights: $\mathcal{P}(\mathcal{D}|\mathcal{M}_m) = \int \mathcal{P}(\mathcal{D}|\mathbf{w}, \mathcal{M}_m)\, \mathcal{P}(\mathbf{w}|\mathcal{M}_m)\, d\mathbf{w}$.[7] It is common for the posterior

[7] In the cases that the integral cannot be computed analytically, Monte Carlo sampling methods (Gelman et al.,

$\mathcal{P}(\mathbf{w}|\mathcal{D}, \mathcal{M}_m) \propto \mathcal{P}(\mathcal{D}|\mathbf{w}, \mathcal{M}_m)\mathcal{P}(\mathbf{w}|\mathcal{M}_m)$ to have a strong peak at the most probable weights $\mathbf{w}_{\mathrm{MP}}$ in many problems. Then the posterior distribution could be approximated by *Laplacian approximation*, i.e., second-order Taylor-expansion of the log posterior:

$$\mathcal{P}(\mathbf{w}|\mathcal{D}, \mathcal{M}_m) \cong \mathcal{P}(\mathbf{w}_{\mathrm{MP}}|\mathcal{D}, \mathcal{M}_m) \cdot \exp\left(-\frac{1}{2}(\mathbf{w} - \mathbf{w}_{\mathrm{MP}})^T \cdot \mathbf{H} \cdot (\mathbf{w} - \mathbf{w}_{\mathrm{MP}})\right) \tag{1.13}$$

where the Hessian $\mathbf{H} = -\nabla_{\mathbf{w}}\nabla_{\mathbf{w}} \log \mathcal{P}(\mathbf{w}|\mathcal{D}, \mathcal{M}_m)|_{\mathbf{w}=\mathbf{w}_{\mathrm{MP}}}$. If $\mathbf{w}$ is a W-dimensional vector, and if the posterior is well approximated by the Gaussian (1.13), the evidence can be approximated by multiplying the best fit likelihood by the Occam's factor as follows[8]

$$\underbrace{\mathcal{P}(\mathcal{D}|\mathcal{M}_m)}_{\text{Evidence}} \cong \underbrace{\mathcal{P}(\mathcal{D}|\mathbf{w}_{\mathrm{MP}}, \mathcal{M}_m)}_{\text{Best fit likelihood}} \cdot \underbrace{\mathcal{P}(\mathbf{w}_{\mathrm{MP}}|\mathcal{M}_m) \cdot (2\pi)^{\frac{W}{2}} (\det \mathbf{H})^{-\frac{1}{2}}}_{\text{Occam's factor}}. \tag{1.14}$$

Here, the Occam's factor is obtained from the normalization factor of the Gaussian (1.13) and the prior probability at $\mathbf{w}_{\mathrm{MP}}$. Typically, a complex model with many free parameters will be penalized with a smaller Occam's factor than a simpler model. The Occam's factor is therefore a measure of the model complexity. Which model achieves the greatest evidence is determined by a tradeoff between minimizing this natural complexity measure and minimizing the data misfit.

So far, we have introduced two inductive principles for learning from finite samples that provide different quantitative formulation of Occam's razor. Constructive implementations of these inductive principles bring into being various learning techniques.

1.3 Modern Techniques

In modern techniques for supervised learning, support vector machines are computationally powerful, while Gaussian processes provide promising non-parametric Bayesian approaches. We will introduce the two techniques in two subsections separately.

1995) can be used to approximate the integral. These work by constructing a Markov chain whose equilibrium distribution is the desired distribution $\mathcal{P}(\mathbf{w}|\mathcal{M}_m)$, and the integral is then approximated using samples from the Markov chain. More complicated schemes sampling from the posterior (Neal, 1997a) are necessarily applied in practice.

[8] This formulation could be derived as follows. Taking integration over $\mathbf{w}$ on the both sides (1.13), we obtain $\int \mathcal{P}(\mathbf{w}|\mathcal{D}, \mathcal{M}_m)d\mathbf{w} = 1 \cong \mathcal{P}(\mathbf{w}_{\mathrm{MP}}|\mathcal{D}, \mathcal{M}_m) \cdot (2\pi)^{\frac{W}{2}} (\det \mathbf{H})^{-\frac{1}{2}}$, where we notice that $\mathcal{P}(\mathbf{w}_{\mathrm{MP}}|\mathcal{D}, \mathcal{M}_m) = \mathcal{P}(\mathcal{D}|\mathbf{w}_{\mathrm{MP}}, \mathcal{M}_m) \cdot \mathcal{P}(\mathbf{w}_{\mathrm{MP}}|\mathcal{M}_m)/\mathcal{P}(\mathcal{D}|\mathcal{M}_m)$. By moving the evidence term $\mathcal{P}(\mathcal{D}|\mathcal{M}_m)$ to the left-side, we can then obtain the formulation (1.14).

1.3.1 Support Vector Machines

In the early 1990s, Vapnik and his coworkers invented a computationally powerful class of supervised learning networks, called support vector machines (SVMs) for solving pattern recognition (Boser et al., 1992). The new algorithm design is firmly grounded in the framework of statistical learning theory developed by Vapnik, Chervonenkis and others (Vapnik, 1995), in which VC dimension (Vapnik and Chervonenkis, 1971) provides a measure for the capacity of a neural network to learn from a set of samples. The basic idea of Vapnik's theory is closely related to regularization, nevertheless capacity control is employed for model selection. Later, SVMs were adapted to tackle density estimate and regression (Vapnik, 1998).

A novel loss function with insensitive zone, known as the ϵ-insensitive loss function (ϵ-ILF), has been proposed for regression problems by Vapnik (1995). ϵ-ILF is defined as

$$\ell_\epsilon(y, f(x;\Theta)) = \begin{cases} 0 & if \quad |y - f(x;\Theta)| \leq \epsilon \\ |y - f(x;\Theta)| - \epsilon & otherwise \end{cases}$$

where $\epsilon \geq 0$. The loss is equal to zero if the absolute value of the deviation of the regression function output $f(x;\Theta)$ from the target y is less than ϵ, and it is equal to the absolute value of the deviation minus ϵ otherwise. SVMs for regression (SVR) exploits the idea of mapping input data into a high dimensional RKHS (often infinite) where a linear regression is performed. In order to estimate $\boldsymbol{f}$ from a given training data set $\mathcal{D}$, classical SVR minimizes the regularized risk functional (1.9) with ϵ-ILF

$$\min_{\boldsymbol{f}\in\text{RKHS}} \mathcal{R}(\boldsymbol{f}) = \frac{1}{n}\sum_{i=1}^{n} \ell_\epsilon(y_i, f(x_i;\Theta)) + \lambda \cdot \|\boldsymbol{f}\|^2_{\text{RKHS}} \tag{1.15}$$

where $\epsilon \geq 0$, $\lambda > 0$ and $\|\boldsymbol{f}\|^2_{\text{RKHS}} = \sum_{\tau=1}^{\infty} \frac{\gamma_\tau^2}{\upsilon_\tau}$. The regression function in the RKHS takes the general form (1.11). In addition, a single constant function $\phi_0(x) = 1$ is introduced as an offset in classical SVR, and then

$$f(x;\Theta) = \sum_{\tau=1}^{\infty} \gamma_\tau \phi_\tau(x) + b \tag{1.16}$$

where the offset $b \in \mathbb{R}$. The constant feature is not considered in the RKHS and therefore it is not penalized in the stabilizer as model complexity (Evgeniou et al., 1999).[9]

The regularized functional can be minimized by solving a convex quadratic programming optimization problem. Two sets of slack variables $\boldsymbol{\xi}$ and $\boldsymbol{\xi}^*$ are introduced: $\xi_i \geq y_i - b - f(x_i) - \epsilon$

[9] The offset term could also be encapsulated into the kernel function.

and $\xi_i^* \geq f(x_i)+b-y_i-\epsilon\ \forall i$. The regularized functional (1.15) could be rewritten as the following equivalent optimization problem, which we refer to as the *primal* problem:

$$\min_{\boldsymbol{\gamma},\, b,\, \boldsymbol{\xi},\, \boldsymbol{\xi}^*} C \cdot \sum_{i=1}^{n}(\xi_i + \xi_i^*) + \frac{1}{2}\sum_{\tau=1}^{\infty}\frac{\gamma_\tau^2}{\upsilon_\tau}$$

$$\text{subject to } \begin{cases} y_i - b - \displaystyle\sum_{\tau=1}^{\infty}\gamma_\tau\phi_\tau(x_i) \leq \epsilon + \xi_i \\ \displaystyle\sum_{\tau=1}^{\infty}\gamma_\tau\phi_\tau(x_i) + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0\ \forall i \end{cases} \tag{1.17}$$

where $C = \frac{1}{2n\lambda}$. To solve the optimization problem with constraints of inequality type we have to find a saddle point of the Lagrange functional

$$\begin{aligned} L(\gamma, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*; \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\eta}, \boldsymbol{\eta}^*) &= C \cdot \sum_{i=1}^{n}(\xi_i + \xi_i^*) + \frac{1}{2}\sum_{\tau=1}^{\infty}\frac{\gamma_\tau^2}{\upsilon_\tau} - \sum_{\tau=1}^{n}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\ -\sum_{i=1}^{\infty}\alpha_i\left(\epsilon + \xi_i - y_i + b + \sum_{\tau=1}^{\infty}\gamma_\tau\phi_\tau(x_i)\right) &- \sum_{i=1}^{\infty}\alpha_i^*\left(\epsilon + \xi_i^* + y_i - b - \sum_{\tau=1}^{\infty}\gamma_\tau\phi_\tau(x_i)\right) \end{aligned} \tag{1.18}$$

where α_i, α_i^*, η_i, $\eta_i^* \geq 0\ \forall i$ are the Lagrange multipliers corresponding to the inequalities in the *primal* problem. Maximization with respect to $\boldsymbol{\gamma}$, b, $\boldsymbol{\xi}$ and $\boldsymbol{\xi}^*$ implies the following three conditions:

$$\gamma_\tau = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\upsilon_\tau\phi_\tau(x_i) \tag{1.19}$$

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \tag{1.20}$$

$$\alpha_i + \eta_i = C \text{ and } \alpha_i^* + \eta_i^* = C\ \ \forall i \tag{1.21}$$

Let us we collect all terms involving ξ_i or ξ_i^* in the Lagrangian (1.18) respectively, and these terms could be grouped into two terms, $(C - \eta_i - \alpha_i)\xi_i$ and $(C - \eta_i^* - \alpha_i^*)\xi_i^*$. Due to the KKT condition (1.21), these terms vanish. All terms involving b in (1.18) are $b\sum_{i=1}^{n}(\alpha_i - \alpha_i^*)$, which will vanish due to the KKT condition (1.20). Together with (1.10) and the KKT condition (1.19), the remaining terms in the Lagrangian yield the *dual* optimization problem:

$$\max_{\boldsymbol{\alpha},\, \boldsymbol{\alpha}^*} -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j) + \sum_{i=1}^{n}y_i(\alpha_i - \alpha_i^*) - \epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) \tag{1.22}$$

subject to $\displaystyle\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$, $0 \leq \alpha_i \leq C$ and $0 \leq \alpha_i^* \leq C\ \ \forall i$.

Obviously, the dual optimization problem (1.22) is a constrained quadratic programming

problem. There are lots of "general purpose" algorithms as well as softwares available in the optimization literature for quadratic programming. But they are not very suitable for solving (1.22) because they cannot take the linear constraint $\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$ and the implicit constraint $\alpha_i \cdot \alpha_i^* = 0$ into account effectively.[10] Special designs on the numerical solution for support vector classifier can be adapted to solve (1.22). The idea to fix the size of active variables at two is known as Sequential Minimal Optimization (SMO), which was first proposed by Platt (1999) for support vector classifier design. The merit of this idea is that the sub-optimization problem can be solved analytically. Smola and Schölkopf (1998) applied Platt's SMO for classifier to regression. Later, Keerthi et al. (2001) put forward improvements on SMO for classifier. Shevade et al. (2000) adapted these improvements into the regression algorithm by introducing two thresholds to determine the bias. Keerthi and Gilbert (2002) proved the convergence of SMO algorithm for SVM classifier design. Joachims (1998) proposed SVM$^{\text{Light}}$ that is a general decomposition algorithm for classifier. Laskov (2000) proposed a decomposition method for regression with working set selection principles. SVM$^{\text{Torch}}$ in Collobert and Bengio (2001) adapted the idea to tackle large-scale regression problems, and Lin (2001) proved the asymptotic convergence for decomposition algorithms. These contributions make SVMs training efficient even on large-scale non-sparse data sets.

The regression function contributed by SVR takes the dual form by introducing (1.19) into (1.16):

$$f(x;\Theta) = \sum_{\tau=1}^{\infty} \gamma_\tau \phi_\tau(x) + b = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(x, x_i) + b \tag{1.23}$$

where the bias b could be obtained as a byproduct in the solution of the *dual* problem. In most of the cases, only some of the Lagrange multipliers, i.e., $(\alpha_i - \alpha_i^*)$ in (1.22), differ from zero at the optimal solution. They define the support vectors (SVs) of the problem. The training samples (x_i, y_i) associated with $|\alpha_i - \alpha_i^*|$ satisfying $0 < |\alpha_i - \alpha_i^*| < C$ are called off-bound SVs, the samples with $|\alpha_i - \alpha_i^*| = C$ are called on-bound SVs, and the samples with $|\alpha_i - \alpha_i^*| = 0$ are called non-SVs. Note that the non-SVs do not involve in the solution representation (1.22). This is usually referred to as the *sparseness* property.

From the regression function (1.23), we can see that SVR belongs to the generalized linear model in Figure 1.1. It is also interesting to compare SVR with RBF. It is possible that they possess the same structure, but their training methods are quite different. SVR enjoy the training via solving a convex quadratic programming problem, in the solution of which the number of

[10] The implicit constraint $\alpha_i \cdot \alpha_i^* = 0$ comes from the fact that α_i and α_i^*, associated with the i-th training sample, are corresponding to the inequality constraints in (1.17) respectively, and only one of the inequalities holds at any time.

hidden neurons and the associated weights are determined uniquely.

The performance of SVR crucially depends on the parameters in kernel function and other two parameters, the tradeoff C (i.e. the regularization parameter λ) and the insensitive zone ϵ in ϵ-ILF. Some generalization bounds, such as V_γ dimension (Evgeniou and Pontil, 1999) or entropy numbers on capacity control (Williamson et al., 1998), could provide a principled way to select model. However, most of the generalization bounds in existence are either not tight enough to determine some elemental parameters, or computationally difficult to implement in practice. Currently, cross validation (Wahba, 1990) is widely used in practice to pick the best parameters, though it would appear difficult to be used when a large number of parameters are involved.

1.3.2 Stationary Gaussian Processes

The application of Bayesian techniques to neural networks was pioneered by Burtine and Weigend (1991), MacKay (1992c) and Neal (1992), and reviewed by Bishop (1995), MacKay (1995) and Lampinen and Vehtari (2001). Bayesian techniques for neural networks specify a hierarchical model with a prior distribution over hyperparameters, and specify a prior distribution of the weights relative to the hyperparameters. This induces a posterior distribution over the weights and hyperparameters for a given data set. Neal (1996) observed that a Gaussian prior for hidden-to-output weights results in a Gaussian process prior for functions as the number of hidden units goes to infinity. Inspired by Neal's work, Williams and Rasmussen (1996) extended the use of Gaussian process prior to higher dimensional problems that have been traditionally tackled with other techniques, such as neural networks, decision trees etc., and good results have been obtained. The important advantage of Gaussian process models for supervised learning (Williams, 1998; Williams and Barber, 1998) over other non-Bayesian models is the explicit Bayesian formulation. This not only builds up Bayesian framework to implement hyperparameter inference but also provides us with confidence intervals in prediction.

Assume that we are observing function values $f(x_i)$ at locations x_i. These observed function values $\{f(x_i)|i = 1, \ldots, n\}$ are regarded as the realizations of random variables in a stationary stochastic process. It is natural to assume that the $f(x_i)$ are correlated, depending on their location x_i merely, i.e., the adjacent observations should convey information about each other to some degree. This is the basis on which we would be able to perform inference. In practice, we usually make a further stringent assumption regarding the distribution of the $f(x_i)$. We could of course assume any arbitrary distribution for the $f(x_i)$. For computational convenience, we

assume they are random variables in a stationary Gaussian process, namely that they form a Gaussian distribution with mean μ and a $n \times n$ covariance matrix Σ whose ij-th element is a covariance function $Cov[f(x_i), f(x_j)]$. The covariance function $Cov[f(x_i), f(x_j)]$ is a function of two locations x_i and x_j, i.e. $Cov(x_i, x_j)$, and returns the covariance between the corresponding outputs $f(x_i)$ and $f(x_j)$. Formally, the covariance function $Cov(x_i, x_j)$ is well-defined, symmetric and the covariance matrix Σ is positive definite. In classical settings, the mean is specified at zero, which implies that we have no prior knowledge about the particular value of the estimate, but assume that small values are preferred.[11] Now let us formulate the prior distribution resulting from these assumptions. For the given set of random variables $\boldsymbol{f} = [f(x_1), f(x_2), \ldots, f(x_n)]^T$, an explicit expression of the prior density function $\mathcal{P}(\boldsymbol{f})$ could be given as

$$\mathcal{P}(\boldsymbol{f}) = \frac{1}{(2\pi)^{\frac{n}{2}} (\det \Sigma)^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\ \boldsymbol{f}^T \cdot \Sigma^{-1} \cdot \boldsymbol{f}\right) \tag{1.24}$$

which is a multivariate joint Gaussian.

In regression problems, we can observe target values y_i which is $f(x_i)$ corrupted by additive noise δ_i as in (1.1), rather than observing $f(x_i)$ directly. The additive noise variables δ_i are independent random variables with unknown distribution. We could assume that δ_i are drawn from different distributions, $\delta_i \sim \mathcal{P}_i(\delta_i)$. This follows that we can state the likelihood $\mathcal{P}(\boldsymbol{y}|\boldsymbol{f})$ as

$$\mathcal{P}(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{n} \mathcal{P}_i(\delta_i) = \prod_{i=1}^{n} \mathcal{P}_i\left(y_i - f(x_i)\right)$$

where the vector of random variables $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^T$. The posterior distribution is then given as:[12]

$$\mathcal{P}(\boldsymbol{f}|\boldsymbol{y}) \propto \mathcal{P}(\boldsymbol{y}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f}) = \prod_{i=1}^{n} \mathcal{P}_i\left(y_i - f(x_i)\right) \cdot \frac{1}{(2\pi)^{\frac{n}{2}} (\det \Sigma)^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\ \boldsymbol{f}^T \cdot \Sigma^{-1} \cdot \boldsymbol{f}\right)$$

To perform inference, we have to further specify the noise distribution that connects $\boldsymbol{f}$ and $\boldsymbol{y}$. For computational convenience, we usually assume that all δ_i are drawn from a same distribution, and the distribution is a Gaussian with zero-mean and variance σ^2, i.e., $\delta_i \sim \mathcal{N}(0, \sigma^2)$, $\forall i$. The advantage of this assumption is that all the distributions involved in the process of inference

[11] In regression problems as in (1.1), to prevent the scalar in target values y_i from impairing this assumption, we usually normalize the targets into zero mean and unit variance in pre-processing.

[12] In the notation of these distributions, there is an implicit condition that the input locations $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ are already given.

remain normal, namely conjugate. Thus, the posterior distribution is

$$\mathcal{P}(\boldsymbol{f}|\boldsymbol{y}) \propto \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{y}-\boldsymbol{f}\|^2 \ -\frac{1}{2}\,\boldsymbol{f}^T\cdot\Sigma^{-1}\cdot\boldsymbol{f}\right) \triangleq \exp\left(-\frac{1}{2}(\boldsymbol{f}-\boldsymbol{f}_{\mathrm{m}})^T\big(\Sigma^{-1}+\frac{1}{\sigma^2}\mathbf{I}\big)(\boldsymbol{f}-\boldsymbol{f}_{\mathrm{m}})\right) \tag{1.25}$$

where $\boldsymbol{f}_{\mathrm{m}}$ is a vector of posterior mean. The posterior distribution (1.25) is again Gaussian with the variance $\big(\Sigma^{-1}+\frac{1}{\sigma^2}\mathbf{I}\big)^{-1}$. For normal distributions, the maximum a posteriori (MAP) estimate of $\boldsymbol{f}$ that maximizes the posterior distribution $\mathcal{P}(\boldsymbol{f}|\boldsymbol{y})$, i.e., $\arg\max_{f}\mathcal{P}(\boldsymbol{f}|\boldsymbol{y})$, is identical with the mean $\boldsymbol{f}_{\mathrm{m}}$, which can consequently found by differentiation

$$\left.\frac{\partial\mathcal{P}(\boldsymbol{f}|\boldsymbol{y})}{\partial\boldsymbol{f}}\right|_{\boldsymbol{f}=\boldsymbol{f}_{\mathrm{m}}} = 0 \implies \boldsymbol{f}_{\mathrm{m}} = \Sigma\cdot(\Sigma+\sigma^2\mathbf{I})^{-1}\cdot\boldsymbol{y}$$

Summarily, in standard Gaussian processes for regression (Williams and Rasmussen, 1996), a Gaussian process prior for the function values is combined with a likelihood evaluated by a Gaussian noise model with zero mean and a standard deviation σ^2 that does not depend on the inputs, to yield a posterior over functions that can be computed exactly using matrix operations.

In the prediction using the Gaussian process, we are interested in calculating the distribution of the random variable $f(x)$ indexed by the new test case x. The $n+1$ random variables $\{y_1,\ldots,y_n,f(x)\}$ have a joint Gaussian distribution as follows

$$\mathcal{P}(\boldsymbol{y},f(x)) \propto \exp\left(-\frac{1}{2}\begin{bmatrix}\boldsymbol{y}\\ f(x)\end{bmatrix}^T\begin{bmatrix}\Sigma+\sigma^2\mathbf{I} & \boldsymbol{k}\\ \boldsymbol{k}^T & Cov(x,x)\end{bmatrix}\begin{bmatrix}\boldsymbol{y}\\ f(x)\end{bmatrix}\right)$$

where $\boldsymbol{k}=[Cov(x_1,x),Cov(x_2,x),\ldots,Cov(x_n,x)]^T$ and Σ is the $n\times n$ covariance matrix. Since we have already observed targets $\boldsymbol{y}=[y_1,y_2,\ldots,y_n]^T$, we can obtain the conditional distribution $\mathcal{P}(f(x)|\boldsymbol{y})$ (Anderson, 1958). Let us keep $\boldsymbol{y}$ intact and make a non-singular linear transformation to $f(x)$:

$$\begin{aligned}\boldsymbol{y} &= \boldsymbol{y}\\ f^* &= f(x)+\boldsymbol{y}^T\cdot\mathrm{L}\end{aligned}$$

where L is an unknown column vector. In order to make $\boldsymbol{y}$ and f^* uncorrelated, we set $E[(\boldsymbol{y}-E(\boldsymbol{y}))(f^*-E(f^*))]=0$ that leads to $\boldsymbol{k}+(\Sigma+\sigma^2\mathbf{I})\cdot\mathrm{L}=0$, i.e. $\mathrm{L}=-(\Sigma+\sigma^2\mathbf{I})^{-1}\cdot\boldsymbol{k}$.[13] The mean of f^* is zero, and the variance of f^* is given as

$$E\left[(f(x)-\boldsymbol{y}^T(\Sigma+\sigma^2\mathbf{I})^{-1}\boldsymbol{k})^T\cdot(f(x)-\boldsymbol{y}^T(\Sigma+\sigma^2\mathbf{I})^{-1}\boldsymbol{k})\right] = Cov(x,x)-\boldsymbol{k}^T(\Sigma+\sigma^2\mathbf{I})^{-1}\boldsymbol{k}$$

[13] On the basis of our assumption of zero-mean Gaussian processes, we know that $E[\boldsymbol{y}]=\mathbf{0}$, $E[f(x)]=0$, $E[\boldsymbol{y}\cdot f(x)]=\boldsymbol{k}$, $E[f(x)\cdot f(x)]=Cov(x,x)$, and $E[\boldsymbol{y}\cdot\boldsymbol{y}^T]=(\Sigma+\sigma^2\cdot\mathbf{I})^{-1}$.

Therefore, $\mathcal{P}(f(x)|\boldsymbol{y})$ is a Gaussian distribution with

$$\begin{aligned} E_{f(x)|\boldsymbol{y}}[f(x)] &= \boldsymbol{y}^T(\Sigma + \sigma^2\mathbf{I})^{-1}\boldsymbol{k} \\ Var_{f(x)|\boldsymbol{y}}[f(x)] &= Cov(x,x) - \boldsymbol{k}^T(\Sigma + \sigma^2\mathbf{I})^{-1}\boldsymbol{k} \end{aligned} \tag{1.26}$$

i.e., $\mathcal{P}(f(x)|\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{y}^T(\Sigma + \sigma^2\mathbf{I})^{-1}\boldsymbol{k}, Cov(x,x) - \boldsymbol{k}^T(\Sigma + \sigma^2\mathbf{I})^{-1}\boldsymbol{k}\right)$.

Given a covariance function, it is straightforward to make a linear combination of the observational targets as the prediction for new test points. However, we are unlikely to know which covariance function to use in practical situations. Thus, it is necessary to choose a parametric family of covariance function (Neal, 1997a; Williams, 1998). We collect the parameters in covariance function as θ, the hyperparameter vector, and then either to estimate these hyperparameters θ via type II maximum likelihood or to use Bayesian approaches in which a posterior distribution over these hyperparameters θ is obtained. This calculations are facilitated by the fact that the distribution $\mathcal{P}(\boldsymbol{y}|\theta)$ can be formulated analytically as

$$\mathcal{P}(\boldsymbol{y}|\theta) = \frac{1}{(2\pi)^{\frac{n}{2}}(\det(\Sigma + \sigma^2\mathbf{I}))^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\boldsymbol{y}^T(\Sigma + \sigma^2\mathbf{I})^{-1}\boldsymbol{y}\right) \tag{1.27}$$

where Σ and σ are functions of θ.

The probability $\mathcal{P}(\boldsymbol{y}|\theta)$ expresses how likely we observe the target values $\{y_1, y_2, \ldots, y_n\}$ as the realization of random variables $\boldsymbol{y}$ if θ is given. Thus, the probability $\mathcal{P}(\boldsymbol{y}|\theta)$ is the likelihood of θ, which is also called the evidence of θ popularly. Since the logarithm is monotonic, likelihood maximization is equivalent to minimize the negative log likelihood $\mathcal{L}(\theta) = -\ln \mathcal{P}(\boldsymbol{y}|\theta)$, which can be given as

$$\mathcal{L}(\theta) = -\ln \mathcal{P}(\boldsymbol{y}|\theta) = \frac{1}{2}\ln\det(\Sigma + \sigma^2\mathbf{I}) + \frac{1}{2}\boldsymbol{y}^T(\Sigma + \sigma^2\mathbf{I})^{-1}\boldsymbol{y} + \frac{n}{2}\ln 2\pi \tag{1.28}$$

It is also possible to analytically express the partial derivatives of the log likelihood with respect to the hyperparameters, using the shorthand $\mathbf{H} = \Sigma + \sigma^2\mathbf{I}$, as

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \frac{1}{2} trace\left(\mathbf{H}^{-1}\frac{\partial \mathbf{H}}{\partial \theta}\right) - \frac{1}{2}\boldsymbol{y}^T\mathbf{H}^{-1}\frac{\partial \mathbf{H}}{\partial \theta}\mathbf{H}^{-1}\boldsymbol{y} \tag{1.29}$$

Note that the evaluation of the likelihood and the partial derivatives takes time $\mathcal{O}(n^3)$, since it involves the inversion of $\mathbf{H}$, which is a $n \times n$ matrix. This is a heavy computational burden for large data sets.

Given $\mathcal{L}(\theta)$ (1.28) and its gradients (1.29), the standard gradient-based optimization pack-

ages could be applied to update these hyperparameters towards a minimum of $-\ln \mathcal{P}(\boldsymbol{y}|\theta)$, i.e., a maximum of the likelihood (1.27). This is the type II maximum likelihood approach for hyperparameter adaptation.

In general, some hyperparameters could be poorly determined in the maximum likelihood estimation when there might be local minima in the likelihood surface. We may concern about the uncertainty in hyperparameter inference while making predictions. The full Bayesian treatment is attractive for erasing the uncertainty. A prior distribution over the parameters $\mathcal{P}(\theta)$ is required to be specified and then a posterior distribution once the target data $\boldsymbol{y}$ has been given $\mathcal{P}(\theta|\boldsymbol{y})$ could be obtained as $\mathcal{P}(\theta|\boldsymbol{y}) \propto \mathcal{P}(\boldsymbol{y}|\theta)\mathcal{P}(\theta)$. While making a prediction for a new test case x, we simply average over the posterior distribution $\mathcal{P}(\theta|\boldsymbol{y})$, i.e.,

$$\mathcal{P}(f(x)|\boldsymbol{y}) = \int \mathcal{P}\big(f(x)|\theta, \boldsymbol{y}\big)\, \mathcal{P}(\theta|\boldsymbol{y})\, d\theta \tag{1.30}$$

It is not possible to do this integration analytically in general, but hybrid Monte Carlo (MCMC) methods (Neal, 1997a) can be used to approximate the integral by using the gradients of $\mathcal{P}(\boldsymbol{y}|\theta)$ to choose search directions which favor the regions of high posterior probability of θ.

1.4 Motivation

Support vector machines for regression (SVR), as an elegant tool for regression problem, 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: a global minimum solution as the minimization of a convex programming problem; relatively fast training speed; and sparseness in solution representation. However, 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, are commonly used in practice to decide values of these hyperparameters, but such approaches are very expensive when a large number of parameters are involved. Typically, Bayesian methods are regarded as suitable tools to determine the values of these hyperparameters.

The important advantage of regression with Gaussian processes (GPR) over other non-Bayesian models is the explicit probabilistic formulation. This not only builds the ability to infer hyperparameters in Bayesian framework but also provides confidence intervals in prediction. However, the inversion of the covariance matrix, whose size is equal to the number of

training samples, must be carried out when the hyperparameters are being adapted. The computational cost of this approach for large data set is very expensive. This drawback of GPR models makes it difficult to deal with over one thousand training samples.

For every RKHS there corresponds a zero-mean stationary Gaussian process with the covariance function defined by the reproducing kernel. The duality between RKHS and stochastic processes is known as the Isometric Isomorphism Theorem (Parzon, 1970; Wahba, 1990). Therefore, with the assumption that a priori $\mathcal{P}(\boldsymbol{f}) \propto \exp(-\lambda\|\boldsymbol{f}\|^2_{\mathrm{RKHS}})$ and the likelihood $\mathcal{P}(\mathcal{D}|\boldsymbol{f}) \propto \exp(-\sum_{i=1}^{n} \ell_\epsilon(y_x, f(x_i;\Theta)))$, the minimizer of the SVR regularized functional (1.15) could be directly interpreted as maximum a posteriori estimate of the function $\boldsymbol{f}$ in the RKHS (Evgeniou et al., 1999). The function $\boldsymbol{f}$ could also be explained as a family of random variables in a Gaussian process due to the Isometric Isomorphism Theorem.

Our intention is to integrate support vector machines with Gaussian processes tightly, while preserving their individual advantages as more as possible. Hence, the contributions of this work might be two-fold: for classical support vector machines, we apply the standard Bayesian techniques to implement model selection, which is convenient to tune large number of hyperparameters automatically; for standard Gaussian processes, we introduce sparseness into Bayesian computation that helps to reduce the computational cost and makes it possible to tackle reasonably large-scale data sets.

1.5 Organization of This Thesis

In this thesis, we focus on regression problems in the first four chapters, and then in Chapter 5 extend our discussion to binary classification problems. The thesis is organized as follows: in Chapter 2 we review the popular loss functions, and then propose soft insensitive loss function as a unified loss function and describe some of its useful properties; in Chapter 3 we review Bayesian designs on generalized linear models that include Bayesian neural networks and Gaussian processes; a detailed Bayesian design for support vector regression is discussed in Chapter 4; we put forward a Bayesian design for binary classification problems in Chapter 5 and we conclude the thesis in Chapter 6.

Chapter 2

Loss Functions

The most general and complete description of the generator of the data is in terms of the probability distribution $\mathcal{P}(x, y)$ in the joint input-target space. For regression problems, it is convenient to decompose the joint probability into the product of the conditional density of the target y, conditioned on the input x, and the unconditional density of the input x, i.e.,

$$\mathcal{P}(x, y) = \mathcal{P}(y|x)\mathcal{P}(x), \tag{2.1}$$

where $\mathcal{P}(y|x)$ denotes the probability density function of y given the fact that x takes a particular value, while $\mathcal{P}(x)$ represents the unconditional density of x. Although the density $\mathcal{P}(x)$ plays an important role in the joint distribution, for the purpose of making prediction of y for new input location x, it is only the conditional distribution of the output variable y, conditioned on x, i.e., $\mathcal{P}(y|x)$, which we need to model. The distribution $\mathcal{P}(x)$ is not taken into account in the modelling process. As a framework for modelling the conditional probability density $\mathcal{P}(y|x)$, Bayesian neural networks or Gaussian processes can yield the distribution of the target y when the trained framework is subsequently presented with a new value for the input vector x.

In the modelling (or training) process, the likelihood could be generally defined as

$$\mathcal{L} = \prod_{i=1}^{n} \mathcal{P}(x_i, y_i) = \prod_{i=1}^{n} \mathcal{P}(y_i|x_i)\mathcal{P}(x_i), \tag{2.2}$$

where we have assumed that each pair of data (x_i, y_i) is drawn independently from the same distribution. Instead of maximizing the likelihood directly, it is generally more convenient to

minimize the negative logarithm of the likelihood. We therefore minimize

$$-\ln \mathcal{L} = -\sum_{i=1}^{n} \ln \mathcal{P}(y_i|x_i) - \sum_{i=1}^{n} \ln \mathcal{P}(x_i). \tag{2.3}$$

As we have mentioned, a Bayesian neural network or a Gaussian process is the model for $\mathcal{P}(y|x)$. The second term in the right-hand side of (2.3) does not depend on the model parameters, and so represents an additive constant which could be dropped from the negative logarithm of the likelihood. For this reason, we can simply state

$$-\ln \mathcal{L} = -\sum_{i=1}^{n} \ln \mathcal{P}(y_i|x_i). \tag{2.4}$$

Note that (2.4) takes the form of a sum over all patterns of an error term for each patterns. This comes from the assumed independence of the data points in the given distribution.

For regression problems, the conditional distribution $\mathcal{P}(y_i|x_i)$ is equivalent to the distribution of the additive noise in measurement, i.e., the δ_i in (1.1). The likelihood about model parameters is essentially a model of the noise, and if the additive noise δ_i is i.i.d. with common probability distribution $\mathcal{P}(\delta)$, it can be written as:

$$\prod_{i=1}^{n} \mathcal{P}(y_i|x_i) = \prod_{i=1}^{n} \mathcal{P}\big(y_i - f(x_i;\Theta)\big) = \prod_{i=1}^{n} \mathcal{P}(\delta_i), \tag{2.5}$$

where $f(x_i;\Theta)$ is the output given by the regression function at the input location x_i and Θ denotes the set of the parameters of the regression function. Furthermore, any $\mathcal{P}(\delta)$ can always be written in the exponential form

$$\mathcal{P}(\delta) = \frac{1}{\mathcal{Z}(C)} \exp\big(-C \cdot \ell(\delta)\big), \tag{2.6}$$

where $\ell(\delta)$ is called the loss function, C is a parameter greater than zero and the normalization factor $\mathcal{Z}(C) = \int \exp\left(-C \cdot \ell(\delta)\right) d\delta$. Note that with this notation, (2.4) can be also written as the sum of loss functions over patterns[1]

$$-\ln \mathcal{L} = C \sum_{i=1}^{n} \ell(\delta_i) + n \ln \mathcal{Z}. \tag{2.7}$$

Thus, different choices of the loss function arise from various assumptions about the distribution

[1] We could assume different distribution $\mathcal{P}_i(\delta_i) = \frac{1}{\mathcal{Z}_i(C_i)} \exp\big(-C_i \cdot \ell(\delta_i)\big)$ for the additive noise δ_i, and then we have $-\ln \mathcal{L} = \sum_{i=1}^{n} C_i \ell(\delta_i) + \sum_{i=1}^{n} \ln \mathcal{Z}_i(C_i)$. However, it is usually hard to determine the optimal parameter C_i for each sample in practice.

of the additive noise $\mathcal{P}(\delta)$.

2.1 Review of Loss Functions

The assumption about the distribution of the additive noise $\mathcal{P}(\delta)$ equivalently determines the form of the loss function $\ell(\delta)$. Gaussian distribution is the traditional assumption for the noise, which is extensively used due to its nice properties in statistical analysis. The loss function associated with Gaussian distribution is a quadratic loss function, whereas non-quadratic loss functions acquire more attention recently.

2.1.1 Quadratic Loss Function

The most popular assumption about the distribution of the additive noise δ is a Gaussian noise model with zero mean, and a variance σ^2 that does not depend on the inputs, i.e.,

$$\mathcal{P}_G(\delta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\delta^2}{2\sigma^2}\right). \tag{2.8}$$

The loss function corresponding to Gaussian distribution is the quadratic loss function, which can be defined as

$$\ell_q(\delta) = \frac{1}{2}\,\delta^2 = \frac{1}{2}\,\big(y - f(x;\Theta)\big)^2, \tag{2.9}$$

where Θ denotes the set of the parameters of the regression function. The relationship between $\mathcal{P}_G(\delta)$ and $\ell_q(\delta)$ can be given as $\mathcal{P}_G(\delta) \propto \exp\big(-C \cdot \ell_q(\delta)\big)$, where $C = \dfrac{1}{\sigma^2}$. The quadratic loss function, which is also called the L_2 loss function. In the following, we relate some well-known results about the statistical analysis on the learning process.

2.1.1.1 Asymptotical Properties

Consider the limit in which the size n of the training data set goes to infinity. In the limit, we can replace the finite sum over patterns in quadratic loss with an integral as follows

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2}\delta_i^2 = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2}\big(y_i - f(x_i;\Theta)\big)^2 = \frac{1}{2} \int\int \big(y - f(x;\Theta)\big)^2 \mathcal{P}(x,y)\, dy\, dx, \tag{2.10}$$

where we introduce an extra factor $1/n$ into the definition of the quadratic loss in order to make the limiting process meaningful. Note that the integral in (2.10) is just the risk functional $\mathcal{R}(\Theta)$ (1.6) using the quadratic loss function (2.9).

We now factor the joint distribution $\mathcal{P}(x, y)$ into the product of the unconditional density function for the input data $\mathcal{P}(x)$, and the conditional density function of target data on the input vector $\mathcal{P}(y|x)$, to give

$$\mathcal{R}(\Theta) = \frac{1}{2} \int \int \big(y - f(x;\Theta)\big)^2 \mathcal{P}(y|x)\mathcal{P}(x)\, dy\, dx, \tag{2.11}$$

Let us define the following conditional averages of the target data

$$E[y|x] = \int y\, \mathcal{P}(y|x)\, dy, \tag{2.12}$$

$$E[y^2|x] = \int y^2\, \mathcal{P}(y|x)\, dy. \tag{2.13}$$

We can write the loss term in (2.11) in the form

$$\begin{aligned} \big(y - f(x;\Theta)\big)^2 &= \big(y - E[y|x] + E[y|x] - f(x;\Theta)\big)^2 \\ &= (y - E[y|x])^2 + 2(y - E[y|x])\big(E[y|x] - f(x;\Theta)\big) \\ &\quad + \big(E[y|x] - f(x;\Theta)\big)^2. \end{aligned} \tag{2.14}$$

Next we substitute (2.14) into (2.11) and make use of (2.12) and (2.13). The second term of (2.14) then vanishes as a consequence of the integration over y. The risk functional (2.11) can then be written in the form

$$\mathcal{R}(\Theta) = \frac{1}{2} \int \big(f(x;\Theta) - E[y|x]\big)^2 \mathcal{P}(x)\, dx + \frac{1}{2} \int \big(E[y^2|x] - (E[y|x])^2\big)\mathcal{P}(x)\, dx. \tag{2.15}$$

We now note that the second term in (2.15) is independent of the regression function $f(x;\Theta)$. For the purpose of modelling the regression function by risk minimization, this term can be ignored. Since the integral in the first term of (2.15) is nonnegative, the minimum of the risk function occurs when the first term vanishes, which corresponds to the following result about the regression function

$$f(x;\Theta^*) = E[y|x] \tag{2.16}$$

where Θ^* is the set of free parameters at the minimum of the risk function. This is a key result and says that the regression function should be given as the conditional average of the target data conditioned on x. Another important result could be obtained when we notice that the

second term in (2.15) can be written in the form

$$\frac{1}{2}\int \sigma^2(x)\mathcal{P}(x)\,dx, \tag{2.17}$$

where $\sigma^2(x)$ represents the variance of the target data, as a function of x, and is defined as

$$\sigma^2(x) = E[y^2|x] - (E[y|x])^2 = \int (y - E[y|x])^2\mathcal{P}(y|x)\,dy. \tag{2.18}$$

The variance of the target data essentially comes from the variance of the additive noise. In regression problems as defined in (1.1), we usually assume that the target data are collected by $y = f(x) + \delta$ where the additive noise δ is regarded as a zero-mean random variable, and the function $f(x)$ is an x-dependent value. The target y, as the sum of $f(x)$ and δ, is a random variable with an x-dependent mean $f(x)$ and a variance of δ. In the case that the optimal regression function is chosen as the conditional expectation (2.16), the first term in (2.15) vanishes, and then the residual risk is given by (2.17). The value of the residual risk is a measure of the average variance of the target data, which is also equivalent to the estimate on the variance of the additive noise as the size of the training data goes to infinity.

Before we go further to discuss the consequences of these important results, we emphasize that what we have obtained are dependent on two key assumptions. First, the data set must be sufficiently large that it could approximate an infinite data set. Second, the model of regression functions $f(x;\Theta)$ must be sufficiently general that there exists a choice of free parameters Θ which makes the first term in (2.15) sufficiently small. The second assumption would be easily satisfied if universal approximators are used for modelling the regression function, but the first assumption is usually not satisfied in a practice situation, since we must deal with the problems arising from finite-size data set. The finiteness of training data brings forth a weakness for maximum likelihood in modelling universal approximators, which is same as the ill-posed problem of the ERM principle we have mention in Section 1.1. The issue arising from modelling on finite data set is also known as *bias/variance dilemma* (Geman et al., 1992). In the following, we consider this issue and then discuss its implications.

2.1.1.2 Bias/Variance Dilemma

Suppose we consider a training set $\mathcal{D}$ consisting of n patterns which we can use to determine the regression function $f(x;\Theta)$. Now consider the whole ensemble of possible data sets, each containing n patterns, and each drawn from the same joint distribution $\mathcal{P}(x,y)$. We have already

argued that the optimal regression function is given by the conditional average $E(y|x)$. The second term in (2.15) represents the intrinsic error because it is independent of the regression function, which could be ignored here. A measure of the effectiveness of $f(x;\Theta)$ as a predictor of the desired one is given by the first term in (2.15), i.e., the integral of the term $\big(f(x;\Theta)-E(y|x)\big)^2$. The value of the quantity will depend on the particular data set $\mathcal{D}$ on which the regression function is trained. We can eliminate the dependency by considering an average over the complete ensemble of data sets, which we write as

$$E_{\mathcal{D}}\big[\big(f(x;\Theta)-E(y|x)\big)^2\big], \tag{2.19}$$

where $E_{\mathcal{D}}[\cdot]$ denotes the expectation, or ensemble average. Let the symbol $E_{\mathcal{D}}[f(x;\Theta)]$ denote the regression function evaluated over the entire ensemble of data sets. Notice that the term $\big(f(x;\Theta)-E(y|x)\big)^2$ can be equivalently rewritten as

$$\big(f(x;\Theta)-E(y|x)\big)^2=\big(f(x;\Theta)-E_{\mathcal{D}}[f(x;\Theta)]+E_{\mathcal{D}}[f(x;\Theta)]-E(y|x)\big)^2, \tag{2.20}$$

where we have simply added and subtracted the average $E_{\mathcal{D}}[f(x;\Theta)]$. By proceeding in a manner similar to that in deriving (2.15), we can decompose the expectation of (2.19) over the ensemble of data sets $\mathcal{D}$ into bias and variance explicitly as follows:

$$\begin{aligned}&E_{\mathcal{D}}\Big[\big(f(x;\Theta)-E(y|x)\big)^2\Big]\\&=\underbrace{\big(E_{\mathcal{D}}[f(x;\Theta)]-E(y|x)\big)^2}_{(\text{bias})^2}+\underbrace{E_{\mathcal{D}}\big[\big(f(x;\Theta)-E_{\mathcal{D}}[f(x;\Theta)]\big)^2\big]}_{\text{variance}}.\end{aligned} \tag{2.21}$$

The expressions for bias and variance in (2.21) are functions of the input vector x. We can introduce the corresponding average values for bias and variance by integrating over all x, so that

$$(\text{bias})^2=\frac{1}{2}\int\big(E_{\mathcal{D}}[f(x;\Theta)]-E(y|x)\big)^2\mathcal{P}(x)\,dx, \tag{2.22}$$

$$\text{variance}=\frac{1}{2}\int E_{\mathcal{D}}\big[\big(f(x;\Theta)-E_{\mathcal{D}}[f(x;\Theta)]\big)^2\big]\mathcal{P}(x)\,dx, \tag{2.23}$$

It is necessary to explain the meaning of the expressions in (2.22) and (2.23). The bias measures the discrepancy between the average result of regression functions over all data sets and the desired function $E(y|x)$. This term represents the inability of the regression function $f(x;\Theta)$ to accurately describe the desired function defined by $E(y|x)$. Conversely, the variance measures the extent to which the regression function is sensitive to the particular choice of data set. To

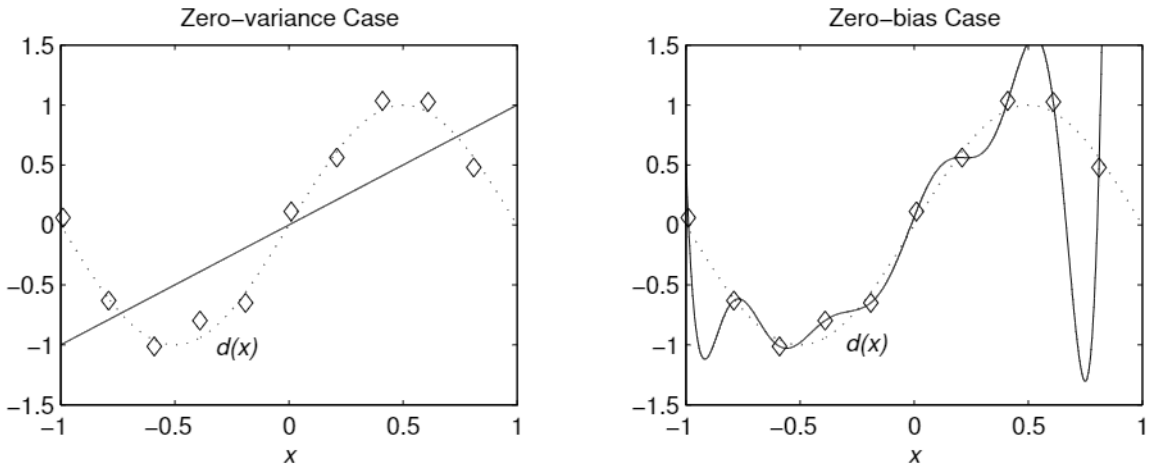

Figure 2.1: Two extreme cases for the choice of regression function $f(x;\Theta)$ to illustrate the bias/variance dilemma. The diamonds denote training samples generated by the underlying model $d(x)$ (dotted curves), and the solid curves denote the regression functions.

achieve good overall performance, the bias term and the variance term of the regression function should both have to be small.

Let us consider two extreme cases for the choice of regression function $f(x;\Theta)$ to illustrate the bias/variance dilemma (Bishop, 1995). We shall suppose that the target data for training is generated from a smooth function $d(x) = \sin(x)$ to which zero mean random variable δ is added, so that $y = d(x) + \delta$. The optimal regression function in this case is given by $E(y|x) = d(x)$. One choice on $f(x;\Theta)$ would be some fixed function $g(x)$ which is completely independent of the data set $\mathcal{D}$, as shown in the left graph of Figure 2.1. It is clear that the variance term (2.23) will vanish, since $E_{\mathcal{D}}[f(x;\Theta)] = g(x) = f(x;\Theta)$. However, the bias term (2.22) will be high since no attention at all was paid to fitting the data. We are making wild guess, unless we have some prior knowledge which helps us choose the regression function $g(x)$. The opposite extreme is to make regression functions which fit the training data perfectly, as indicated in the right graph of Figure 2.1. In this case the bias term vanishes at the data points themselves since that $E_{\mathcal{D}}[f(x;\Theta)] = E_{\mathcal{D}}[d(x) + \delta] = d(x) = E[y|x]$. The variance, however, will be significant, because each regression function heavily depend on its particular training data which have been corrupted by noise, and the variation of their prediction in the neighborhood of the data points will be typically even greater. We see that there is a natural trade-off between bias and variance. A regression function which is complex and has the capability to closely describe the training data set will tend to suffer a large variance and hence give a large expected risk. We can decrease the variance by smoothing the model, but if we go too far then the bias will become large and the

expected risk again large. The analysis on the trade-off between bias and variance is consistent with the principle of Occam's razor, which is the basic principle for model selection and motivates numerous applications in neural networks, such as weight decay (Hinton, 1987), optimal brain damage (LeCun et al., 1990), optimal brain surgeon (Hassibi et al., 1991) and so on.

2.1.1.3 Summary of properties of quadratic loss function

Let us summarize the analysis obtained from the principle of maximum likelihood by assuming that the distribution of the target data could be described by a Gaussian function with an x-dependent mean, and a single global noise variance. In statistics, the optimal regression function should be the conditional average $E(y|x)$. The residual value (2.15) is an estimate on the variance of the additive noise as the size of the training data goes to infinity. Furthermore, there is a trade-off between bias and variance, which is also known as under-fitting for too simple models and over-fitting for too complex models.

In addition, we note that the quadratic loss function does not require that the distribution of target variables or the additive noise be Gaussian. If quadratic loss function is used, the quantities which can be determined in training are the x-dependent mean of the distribution given by the output of the regression function, and the global average noise variance given by the residual value of the risk functional at its minimum. Thus, the quadratic loss function cannot distinguish between the true distribution and the Gaussian distribution with the same x-dependent mean and average variance. This observation indicates that non-quadratic loss functions could also be used in the risk function in place of quadratic loss function to retrieve the x-dependent mean and the noise variance, even when the underlying noise distribution is actually Gaussian.

2.1.2 Non-quadratic Loss Functions

One of the potential difficulties of the standard quadratic loss function is that it receives large contributions from outliers that have particularly large errors. If there are long tails on the 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). Several non-quadratic loss functions have been introduced to reduce the sensitivity to the outliers, such as the Laplacian loss function (Bishop, 1995) and the Huber's loss function.

2.1.2.1 Laplacian Loss Function

If we assume that the additive noise is distributed as $\mathcal{P}_L(\delta) = \frac{C}{2}\exp(-C|\delta|)$, then the loss function is called Laplacian loss function

$$\ell_l(\delta) = |\delta|, \tag{2.24}$$

which is also known as L_1 loss function. With Laplacian loss function, the minimum risk solution computes the conditional *median*[2], rather than the conditional mean. The reason for this can be seen by considering the expectation of $|y - f(x,\Theta)|$ over the distribution $\mathcal{P}(y|x)$. Let us denote c as the median of $\mathcal{P}(y|x)$, and notice that

$$\begin{aligned} E\big[|y - f(x,\Theta)|\big] &= E\big[|y-c|\big] + 2\int_c^{f(x,\Theta)}(c-y)\mathcal{P}(y|x)\,dy \\ &\quad + \big(c - f(x,\Theta)\big)\Big(\mathcal{P}\big(y \geq f(x,\Theta)\big) - \mathcal{P}\big(y < f(x,\Theta)\big)\Big), \end{aligned} \tag{2.25}$$

when $c < f(x,\Theta)$. It is easy to see that the minimum of $E\big[|y - f(x,\Theta)|\big]$ is reached when we choose $f(x,\Theta) = c$. Similar logic applies to the case of $c > f(x,\Theta)$.

We study a simple example of fitting a linear polynomial through a set of noisy data points to illustrate the advantage of linear loss function to outliers, where an extra data point being added artificially lies well away from the other data points, as shown in Figure 2.2. Comparing with the results of the case without outlier, we find that the extra outlier greatly changes the result of quadratic loss function, but slightly influences the result of Laplacian loss function.

2.1.2.2 Huber's Loss Function

Huber's loss function was proposed by Huber (1981) for robust estimators, which is defined as

$$\ell_h(\delta) = \begin{cases} \dfrac{\delta^2}{4\epsilon} & if \;\; |\delta| \leq 2\epsilon \\ |\delta| - \epsilon & otherwise. \end{cases} \tag{2.26}$$

where $\epsilon > 0$. It is a hybrid of quadratic and linear loss functions. The loss is equal to the absolute noise value minus ϵ if the noise value is greater than 2ϵ, and it is quadratic to the noise value otherwise.

[2]For a random variable ζ, the value c satisfying $\mathcal{P}(\zeta \geq c) \geq \frac{1}{2}$ and $\mathcal{P}(\zeta \leq c) \geq \frac{1}{2}$ is called the median of the distribution of ζ.

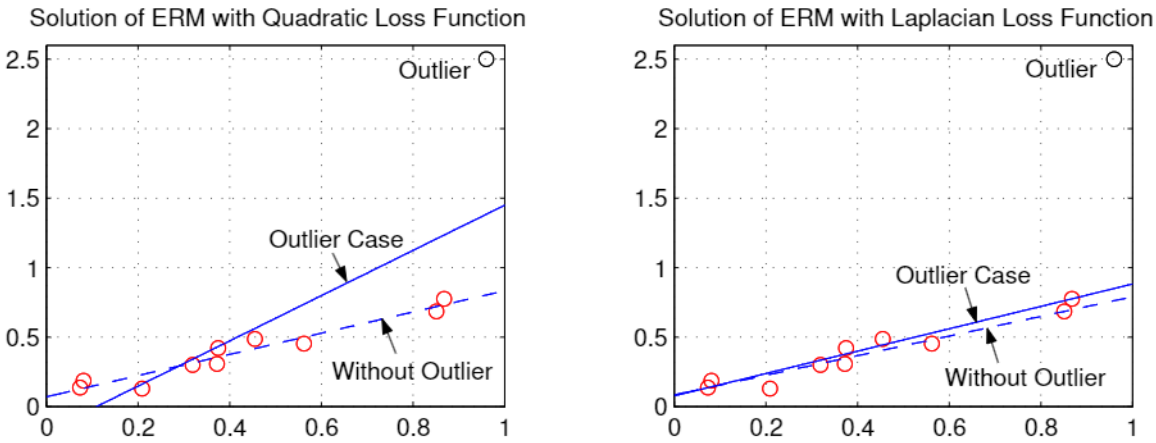

Figure 2.2: An example of fitting a linear polynomial through a set of noisy data points with an outlier.

2.1.2.3 ϵ-insensitive Loss Function

The ϵ-insensitive loss function (ϵ-ILF), introduced by Vapnik (1995), is defined as

$$\ell_\epsilon(\delta) = \begin{cases} 0 & if \ |\delta| \leq \epsilon \\ |\delta| - \epsilon & otherwise. \end{cases} \tag{2.27}$$

It is a linear loss function with a flat zero region. The loss is equal to zero if the absolute noise value is less than ϵ, and it is equal to the absolute noise value minus ϵ otherwise.

From their definitions and Figure 2.3, we notice that Huber's loss function and ϵ-ILF approach the Laplacian loss function as $\epsilon \to 0$. 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 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, as known in SVR. 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 blend their desirable features together and propose a novel loss function, namely soft insensitive loss function, in the next section.

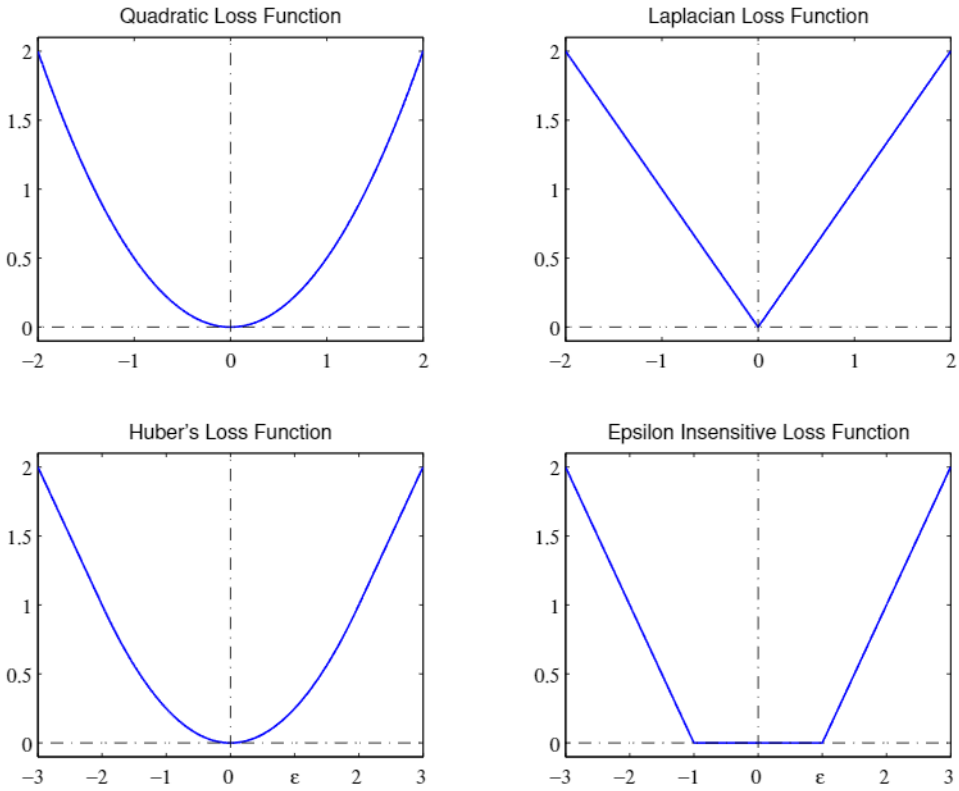

Figure 2.3: Graphs of popular loss functions, where ϵ is set at 1.

2.2 A Unified Loss Function

In this section, we propose a new loss function, namely soft insensitive loss function, as a unified version of the popular loss functions we reviewed in the previous section.

2.2.1 Soft Insensitive Loss Function

The soft insensitive loss function (SILF) (Chu et al., 2001b) is defined as:

$$\ell_{\epsilon,\beta}(\delta) = \begin{cases} -\delta - \epsilon & if \quad \delta \in \Delta_{C^*} = \big(-\infty, -(1+\beta)\epsilon\big) \\ \dfrac{(\delta + (1-\beta)\epsilon)^2}{4\beta\epsilon} & if \quad \delta \in \Delta_{M^*} = [-(1+\beta)\epsilon, -(1-\beta)\epsilon] \\ 0 & if \quad \delta \in \Delta_0 = (-(1-\beta)\epsilon, (1-\beta)\epsilon) \\ \dfrac{(\delta - (1-\beta)\epsilon)^2}{4\beta\epsilon} & if \quad \delta \in \Delta_M = [(1-\beta)\epsilon, (1+\beta)\epsilon] \\ \delta - \epsilon & if \quad \delta \in \Delta_C = \big((1+\beta)\epsilon, +\infty\big) \end{cases} \tag{2.28}$$

where $0 < \beta \leq 1$ and $\epsilon > 0$. The profile of SILF is shown in Figure 2.4. The properties of SILF are entirely controlled by two parameters, β and ϵ. For a fixed ϵ, SILF approaches the ϵ-ILF as

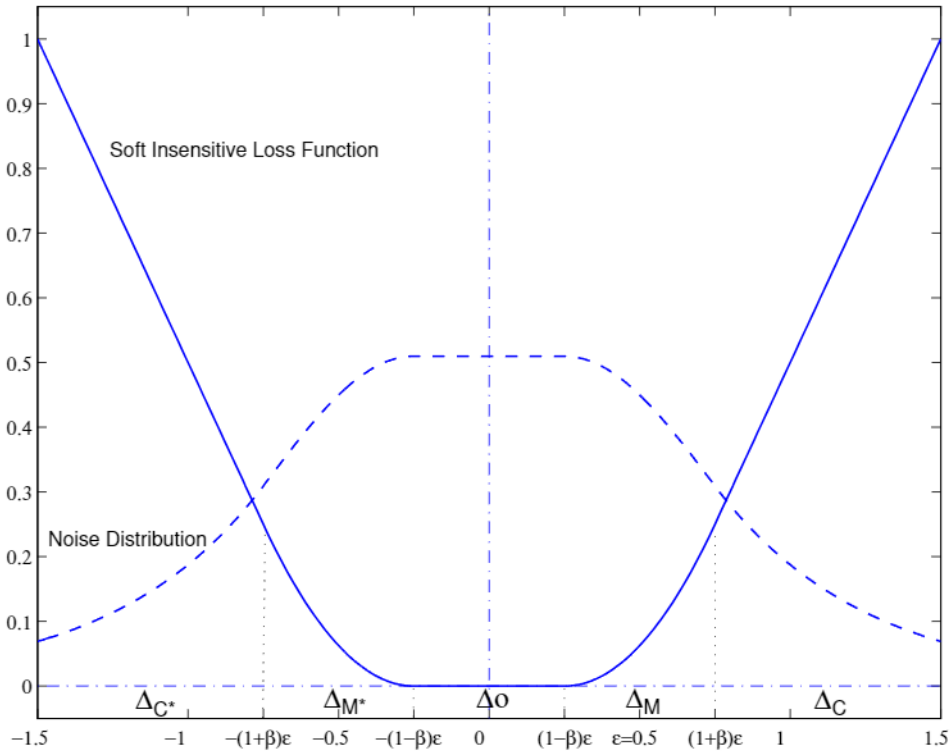

Figure 2.4: Graphs of soft insensitive loss function (solid curve) and its corresponding noise density function (dotted curve), where $\epsilon = 0.5$, $\beta = 0.5$ and $C = 2.0$ in the noise model.

$\beta \to 0$; on the other hand, as $\beta \to 1$, it approaches the Huber's loss function. In addition, SILF becomes the Laplacian loss function as $\epsilon \to 0$. Held ϵ at some large value and let $\beta \to 1$, the SILF approach 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

$$\frac{d\ell_{\epsilon,\beta}(\delta)}{d\delta} = \begin{cases} -1 & if \quad \delta \in \Delta_{C^*} \\ \dfrac{\delta + (1-\beta)\epsilon}{2\beta\epsilon} & if \quad \delta \in \Delta_{M^*} \\ 0 & if \quad \delta \in \Delta_0 \\ \dfrac{\delta - (1-\beta)\epsilon}{2\beta\epsilon} & if \quad \delta \in \Delta_M \\ 1 & if \quad \delta \in \Delta_C \end{cases} \tag{2.29}$$

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

$$\frac{d^2\ell_{\epsilon,\beta}(\delta)}{d\delta^2} = \begin{cases} \dfrac{1}{2\beta\epsilon} & if \quad \delta \in \Delta_{M^*} \cup \Delta_M \\ 0 & otherwise \end{cases} \tag{2.30}$$

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

The density function of the additive noise in measurement corresponding to the choice of SILF is

$$\mathcal{P}_S(\delta) = \frac{1}{\mathcal{Z}_S} \exp\big(-C \cdot \ell_{\epsilon,\beta}(\delta)\big) \tag{2.31}$$

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

$$\mathcal{Z}_S = 2(1-\beta)\epsilon + 2\sqrt{\frac{\pi\beta\epsilon}{C}} \cdot \operatorname{erf}\Big(\sqrt{C\beta\epsilon}\Big) + \frac{2}{C}\exp\big(-C\beta\epsilon\big) \tag{2.32}$$

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

$$\begin{aligned} \sigma_n^2 = \; & \frac{2}{\mathcal{Z}_S}\Bigg\{\frac{(1-\beta)^3\epsilon^3}{3} + \sqrt{\frac{\pi\beta\epsilon}{C}}\left(\frac{2\beta\epsilon}{C} + (1-\beta)^2\epsilon^2\right)\operatorname{erf}\Big(\sqrt{C\beta\epsilon}\Big) \\ & + \frac{4(1-\beta)\beta\epsilon^2}{C} + \left(\frac{\epsilon^2(1-\beta)^2}{C} + \frac{2\epsilon(1+\beta)}{C^2} + \frac{2}{C^3}\right)\exp(-C\beta\epsilon)\Bigg\} \end{aligned} \tag{2.33}$$

2.2.2 A Model of Gaussian Noise

Pontil et al. (1998) have shown that the use of Vapnik's ϵ-insensitive loss function is equivalent to a model of additive and Gaussian noise, where the variance and mean of the Gaussian are i.i.d. random variables. Following their approach, we derive the corresponding Gaussian noise model for SILF now.

If the uncertainties in measurement conditions are taken into account, it seems reasonable to discard the popular assumption that noise variables δ_i are identically distributed. In particular, we assume that the noise variables δ_i have probability distributions $\mathcal{P}_i(\delta_i)$ which are Gaussians, but do not necessarily have zero means and identical variances. Thus, the noise distributions $\mathcal{P}_i(\delta_i)$ can be assumed to be

$$\mathcal{P}_i(\delta_i) = \frac{1}{\sqrt{2\pi}\delta_i} \exp\left(-\frac{(\delta_i - t_i)^2}{2\sigma_i^2}\right) \tag{2.34}$$

where σ_i denotes the standard deviation of the noise associated with the i-th sample and t_i denotes the noise mean of the i-th sample.

Here, we allow for the fact that the noise could be biased in this model, and consider σ_i and t_i as i.i.d. random variables to model the uncertainties in measurement. Therefore, $\mathcal{P}_i(\delta_i)$ can be interpreted as $\mathcal{P}_i(\delta_i|\sigma_i, t_i)$, the conditional probability of δ_i given σ_i and t_i. Then we compute

the marginal of the likelihood (2.5), integrating over $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_n\}$ and $\boldsymbol{t} = \{t_1, \ldots, t_n\}$:

$$\prod_{i=1}^{n} \mathcal{P}_i(\delta_i) = \int d\boldsymbol{\sigma}\, d\boldsymbol{t} \prod_{i=1}^{n} \mathcal{P}_i(\delta_i|\sigma_i, t_i)\mathcal{P}(\boldsymbol{\sigma}, \boldsymbol{t}) \tag{2.35}$$

On the assumption that $\boldsymbol{\sigma}$ and $\boldsymbol{t}$ are i.i.d. random variables, we obtain that

$$\mathcal{P}(\boldsymbol{\sigma}, \boldsymbol{t}) = \prod_{i=1}^{n} \mathcal{P}(\sigma_i, t_i) = \prod_{i=1}^{n} \mu(\sigma_i)\lambda(t_i) \tag{2.36}$$

where $\mu(\cdot)$ and $\lambda(\cdot)$ denote the density distribution of $\boldsymbol{\sigma}$ and $\boldsymbol{t}$ respectively. Finally the likelihood (2.5) can be given as

$$\prod_{i=1}^{n} \mathcal{P}_i(\delta_i) = \prod_{i=1}^{n} \int d\sigma_i\, dt_i \mathcal{P}_i(\delta_i|\sigma_i, t_i)\mu(\sigma_i)\lambda(t_i) \tag{2.37}$$

where $\mathcal{P}_i(\delta_i|\sigma_i, t_i)$ is defined as in (2.34). Let us choose SILF as the loss function. Then, from (2.5) and (2.31), the likelihood can also be written as

$$\prod_{i=1}^{n} \mathcal{P}(y_i|x_i) = \frac{1}{\mathcal{Z}_S^n} \exp\left(-C \sum_{i=1}^{n} \ell_{\epsilon,\beta}(\delta_i)\right) \tag{2.38}$$

where $\ell_{\epsilon,\beta}(\cdot)$ is defined as in (2.28). Since (2.37) and (2.38) are equal, we can easily see that the loss function can also be expressed in the form as follow:

$$\ell_{\epsilon,\beta}(\delta) = -\frac{1}{C} \ln \int_0^{+\infty} d\sigma \int_{-\infty}^{+\infty} dt \frac{\mathcal{Z}_S}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\delta - t)^2}{2\sigma^2}\right) \lambda(t)\mu(\sigma) \tag{2.39}$$

where we drop off the subscript i in σ_i and t_i, as they are identical random variables. Thus, using a loss function with an integral representation of the form (2.39) is equivalent to assuming that the noise is Gaussian, but the mean and the variance of the noise are random variables with individual probability density functions.

2.2.2.1 Density Function of Standard Deviation

We derive the density function for σ in the following. By rearranging the integral in (2.39), we obtain:

$$\exp\big(-C \cdot \ell_{\epsilon,\beta}(\delta)\big) = \int_{-\infty}^{+\infty} dt \lambda(t) G(\delta - t) \tag{2.40}$$

where

$$G(\tau) = \int d\sigma \frac{\mathcal{Z}_S}{\sqrt{2\pi}\sigma} \mu(\sigma) \exp\left(-\frac{\tau^2}{2\sigma^2}\right). \tag{2.41}$$

We notice that the function SILF becomes the Laplacian loss function (2.24) when $\epsilon = 0$. In this case, the noise distribution becomes Laplacian distribution:

$$\mathcal{P}(\delta) = \frac{1}{\mathcal{Z}_S} \exp(-C|\delta|) = \frac{1}{\mathcal{Z}_S} \exp\left(-C \cdot \ell_{0,\beta}(\delta)\right) \tag{2.42}$$

where $\mathcal{Z}_S = \frac{2}{C}$. It is also known that this Laplacian distribution is an unbiased noise distribution, i.e., the density function of its mean t is a delta function at zero (Pontil et al., 1998). Using this fact and the expression in (2.41), (2.40) can be simplified as:

$$\exp\left(-C\ell_{0,\beta}(\delta)\right) = G(\delta) = \int_0^{+\infty} \sqrt{\frac{2}{\pi}} \frac{\mu(\sigma)}{C\sigma} \exp\left(-\frac{\delta^2}{2\sigma^2}\right) d\sigma. \tag{2.43}$$

Such a class of loss function defined by (2.43) is an extension of the model discussed by Girosi (1991). Since $2\sqrt{\pi}\exp(-\sqrt{\tau}) = \int_0^{+\infty} d\eta \exp\left(-\frac{1}{4\eta}\right) \exp(-\tau\eta)\eta^{-\frac{3}{2}}$ holds for any τ, it follows that by setting $\eta = \frac{1}{2C^2\sigma^2}$ and $\tau = C^2\sigma^2$, we get

$$\exp(-C|\delta|) = \int_0^{+\infty} \sqrt{\frac{2}{\pi}}\, C \exp\left(-\frac{C^2\sigma^2}{2}\right) \exp\left(-\frac{\delta^2}{2\sigma^2}\right) d\sigma. \tag{2.44}$$

Comparing (2.44) with (2.43), we find that the density function of the standard deviation is a *Rayleigh* distribution of the form:

$$\mu(\sigma) = C^2\sigma \exp\left(-\frac{C^2\sigma^2}{2}\right) \tag{2.45}$$

2.2.2.2 Density Distribution of Mean

So far, we know $G(\delta) = \exp(-C|\delta|)$ from (2.43) and $\lambda(t)$ is a delta function at zero when $\epsilon = 0$. It remains to obtain the explicit expression of $\lambda(t)$ for $\epsilon > 0$. Taking Fourier transformation on (2.40), we have

$$\tilde{\mathcal{F}}[\exp(-C\ell_{\epsilon,\beta}(\delta))] = \tilde{\lambda}(\omega)\tilde{G}(\omega) \tag{2.46}$$

where $\tilde{\lambda}(\omega)$ and $\tilde{G}(\omega)$ are the Fourier transformation of $\lambda(t)$ and $G(t)$ respectively. From (2.43), we get $\tilde{G}(\omega) = \frac{2C}{C^2 + \omega^2}$. Thus, the Fourier transformation of $\lambda(t)$ shall be

$$\tilde{\lambda}(\omega) = \frac{C^2 + \omega^2}{2C} \tilde{\mathcal{F}}[\exp(-C\ell_{\epsilon,\beta}(\delta))] \tag{2.47}$$

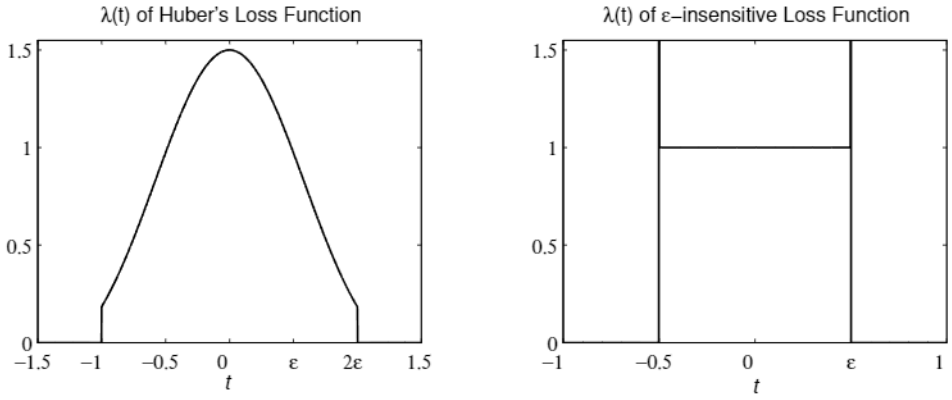

Figure 2.5: Graphs of the distribution on the mean $\lambda(t)$ for Huber's loss function ($\beta = 1.0$) and ϵ-insensitive loss function ($\beta \to 0$), where $\epsilon = 0.5$ and $C = 2.0$.

Using the differential property of $\tilde{\mathcal{F}}(\lambda^{(n)}(t)) = (\mathrm{i}\omega)^n \tilde{\lambda}(\omega)$ and changing the variable δ into t, we find that $\lambda(t)$ is:

$$\lambda(t) = \frac{1}{2C}\big[C^2 \exp\big(-C\ell_{\epsilon,\beta}(t)\big) - \frac{\partial^2}{\partial t^2}\exp\big(-C\ell_{\epsilon,\beta}(t)\big)\big] \tag{2.48}$$

i.e.

$$\lambda(t) = \frac{1}{2}\left[C + \frac{\partial^2 \ell_{\epsilon,\beta}(t)}{\partial t^2} - C\left(\frac{\partial \ell_{\epsilon,\beta}(t)}{\partial t}\right)^2\right] \tag{2.49}$$

From the definitions (2.28), (2.29) and (2.30), the density distribution of mean $\lambda(t)$ can be written in explicit form without normalization as follows:

$$\lambda(t) \propto \begin{cases} 1 & if \quad t \in \Delta_0 \\ \left[1 + \frac{1}{2\beta\epsilon C} - C\left(\frac{|t|-(1-\beta)\epsilon}{2\beta\epsilon}\right)^2\right] & if \quad t \in \Delta_{M^*} \cup \Delta_M \\ 0 & if \quad t \in \Delta_{C^*} \cup \Delta_C \end{cases} \tag{2.50}$$

Finally notice that for the class of loss function, SILF, the noise distribution (2.40) can be written as the convolution between the distributions of the mean $\lambda(t)$ (2.50) and the Laplacian distribution (2.43):

$$\mathcal{P}(\delta) = \int_{-(1+\beta)\epsilon}^{(1+\beta)\epsilon} \lambda(t)\exp(-C|\delta - t|)dt \tag{2.51}$$

The statement (2.51) establishes a representation of the noise distribution $\mathcal{P}(\delta)$ as a continuous superposition of Laplacian functions in the interval $[-(1+\beta)\epsilon, (1+\beta)\epsilon]$.

2.2.2.3 Discussion

In summary, the noise model for SILF can be interpreted as Gaussian, but it could be biased, where the standard deviation and the mean of the Gaussian are i.i.d. random variables with specific probability distributions stated in (2.45) and (2.50) respectively. The parameter C determines the distribution of standard deviation entirely, while parameter ϵ controls main properties of the distribution of the mean. Huber's loss function and ϵ-ILF can be regarded as special cases of SILF that are wholly controlled by the parameter β only. All the three loss functions share the same distribution of standard deviation σ in their noise models, but the distributions of the mean t are different. For a fixed ϵ, as $\beta \to 0$, Δ_{M^*} and Δ_M shrink to points $\pm\epsilon$, and the distribution becomes the delta function at $\pm\epsilon$. Thus, the distribution of mean for ϵ-ILF can be stated as uniform in the interval $(-\epsilon, \epsilon)$ and with two delta functions at $\pm\epsilon$. On the contrary, as $\beta \to 1$, Δ_0 shrinks. When $\beta = 1$, the distribution of mean for Huber's loss function can be stated as

$$\lambda(t) \propto \left(1 + \frac{1}{2\epsilon C} - \left(\frac{t}{2\epsilon}\right)^2\right) \exp\left(-\frac{C \cdot t^2}{4\epsilon}\right) \qquad t \in [-2\epsilon, 2\epsilon] \tag{2.52}$$

The two special cases are presented in Figure 2.5. We find that this formulation (2.50) and the graphs are consistent with the results given by Pontil et al. (1998). Thus, the Gaussian noise model is an extension of the noise model discussed by Pontil et al. (1998).

So far, we have given a brief review of popular loss functions, and also put forward a unified version for the popular loss functions, known as the soft insensitive loss function. The form of the loss function we choose is completely specified by our assumption about the noise distribution, and then the likelihood of the observed data can be evaluated for any given regression function. Together with our prior knowledge, the observed data and the likelihood could be used to refresh our knowledge to the posterior in Bayesian frameworks. In the next chapter, we shall review the Bayesian frameworks for regression, that include Bayesian neural networks and Gaussian processes.

Chapter 3

Bayesian Frameworks

The task of designing a model for a particular system occurs in many areas of research. The system serves as a mapping function, whose inner mechanism is usually unknown. For an input vector, the system yields a unique corresponding output. These pairs of data collected from the system represent the information of the system. We design a mathematical model based on these observational data to represent our beliefs about the system. The model, as a result of learning from data, can then be used to make inference. Figure 3.1 shows a block diagram that illustrates this form of learning, which is usually referred to as *supervised learning*. Here, we need a consistent framework within which to construct our model, incorporating our prior knowledge and within which to consistently update our knowledge of the optimal model. We choose the Bayesian framework in the thesis.

The Bayesian approach is based upon the expression of knowledge in terms of probability distributions. Given the data and a specific model, we can deterministically make inferences using the rules of probability theory. In principle, a unique optimal solution to our data modelling problem exists in the Bayesian framework. However, such a solution may be difficult to find in practice due to prohibitive computational cost. The key to a successful implementation of Bayesian methods is the choice of the model and the mathematical techniques we shall use.

In this chapter, we shall review the Bayesian frameworks for regression that include Bayesian neural networks and Gaussian processes.

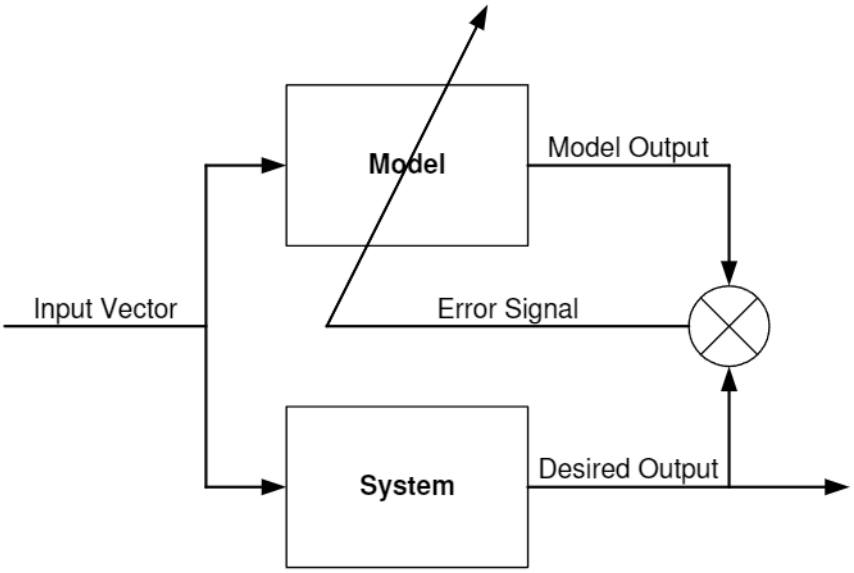

Figure 3.1: Block diagram of supervised learning.

3.1 Bayesian Neural Networks

Neural networks (Bishop, 1995; Haykin, 1999) are widely used in data modelling. The application of Bayesian techniques to back-propagation neural networks was pioneered by Buntine and Weigend (1991), MacKay (1992c), Neal (1992). We focus on the MLP with single hidden layer to relate the basic idea for simplicity. Of course, the Bayesian approach can be extended in a straightforward way to the MLP with any hidden layers. Suppose there is a regression task to model the mapping function $f : \mathbb{R}^d \to \mathbb{R}$ as in (1.1) given a set of training samples $\mathcal{D} = \{(x_i, y_i)|i = 1, \ldots, n\}$. We prepare a set of MLP models with different sizes of hidden neuron $\{\mathcal{M}_m\}$ to account for the training data $\mathcal{D}$ we are given, where m denotes the number of hidden neurons in the MLP model. In Figure 3.2, we present a structural graph for the MLP model $\mathcal{M}_m$ with m hidden neurons. The prior probability $\mathcal{P}(\mathcal{M}_m)$ is assigned to model $\mathcal{M}_m$ which expresses our initial preference on the model $\mathcal{M}_m$ before the observational data arrive. If there is no reason to assign strongly different $\mathcal{P}(\mathcal{M}_m)$, we can simply give an equal value for each model that makes $\sum_m \mathcal{P}(\mathcal{M}_m) = 1$.

Let us collect all the weights as $\mathbf{w}$, the weight vector, and then regard $\mathbf{w}$ as random variables with some probability distribution. In general, we can write this prior distribution in an exponential form

$$\mathcal{P}(\mathbf{w}|\, A) = \frac{1}{\mathcal{Z}_{\mathbf{w}}(A)} \exp(-AE_{\mathbf{w}}) \tag{3.1}$$

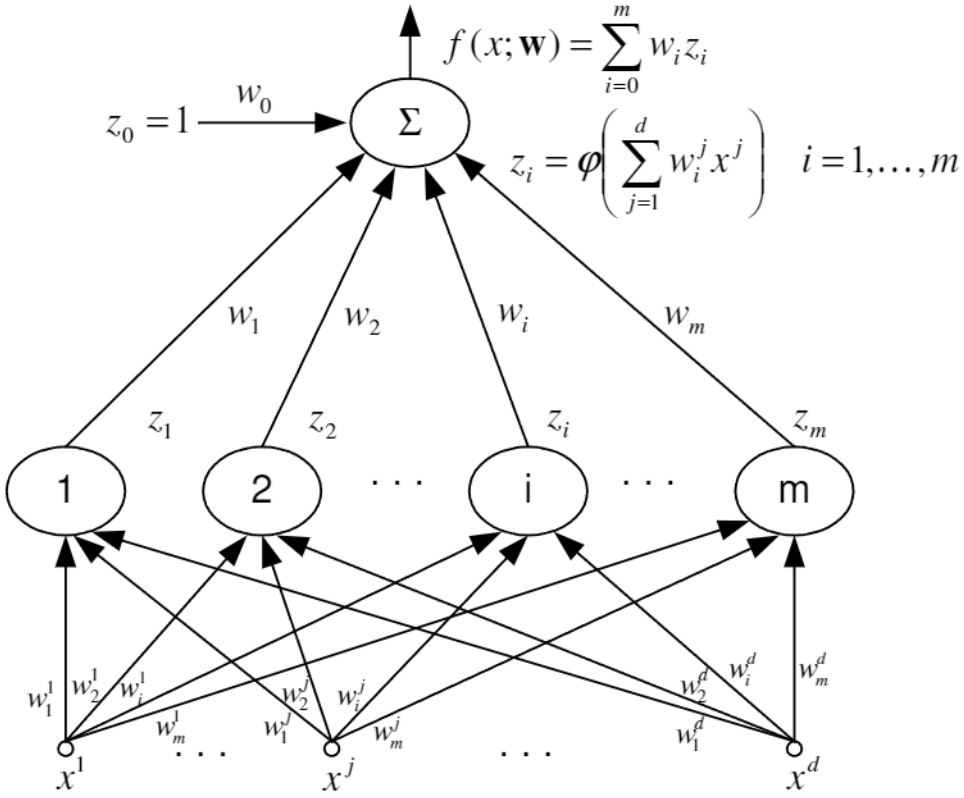

Figure 3.2: The structural graph of MLP with single hidden layer.

where the parameter $A > 0$. Since the parameter A itself controls the distribution of the weight vector $\mathbf{w}$, it is also called a *hyperparameter*. To begin with, we assume that the value of A is known. We shall discuss how to adjust A as part of the modelling process later. For any specified weight vector $\mathbf{w}$, the model's descriptions to the training data $\mathcal{D}$ could be evaluated by a likelihood function, which is essentially a noise model discussed in previous chapter. The likelihood function could be generally written in an exponential form

$$\mathcal{P}(\mathcal{D}|\mathbf{w}, C) = \frac{1}{\mathcal{Z}_{\mathcal{D}}(C)} \exp(-CE_{\mathcal{D}}) \tag{3.2}$$

where the parameter $C > 0$ controls the noise distribution and $\mathcal{Z}_{\mathcal{D}}$ is a normalizing factor. To continue, we assume the value of C is known at present. As another hyperparameter, C will be treated later along with the hyperparameter A. Using the prior distribution in (3.1) and the likelihood function in (3.2), we can write the posterior distribution of the weights in the form,[1]

$$\mathcal{P}(\mathbf{w}|\mathcal{D}, C, A) = \frac{\mathcal{P}(\mathcal{D}|\mathbf{w}, C)\mathcal{P}(\mathbf{w}|A)}{\mathcal{P}(\mathcal{D}|C, A)} \tag{3.3}$$

where the denominator is a normalizing factor which can be written as

$$\mathcal{P}(\mathcal{D}|C, A) = \int \mathcal{P}(\mathcal{D}|\mathbf{w}, C)\mathcal{P}(\mathbf{w}|A)\, d\mathbf{w}. \tag{3.4}$$

which ensures that the left-hand side of (3.3) gives unity when it is integrated over all weight space.

In a Bayesian framework, we also regard the hyperparameters A and C as random variables. To complete the specification of this hierarchical prior, we must define a prior distribution over the hyperparameters A and C. The hyperprior $\mathcal{P}(C, A)$ could be specified subjectively to express our initial idea about the hyperparameter choice before the observational data arrive. As random variables, they are positive and independently distributed. Thus suitable priors could be Gamma distributions (Berger, 1985):

$$\mathcal{P}(A) = \text{Gamma}(A|p_a, q_a) \tag{3.5}$$

$$\mathcal{P}(C) = \text{Gamma}(C|p_c, q_c) \tag{3.6}$$

where $\text{Gamma}(A|p, q) = \Gamma(p)^{-1} q^p A^{p-1} \exp(-qA)$ with $\Gamma(p) = \int_0^\infty t^{p-1} \exp(-t)\, dt$. To make these priors non-informative (i.e. flat distributed), we might fix their parameters to small values. Notice that as an extreme limit of setting these parameters to zero $p_a = q_a = p_c = q_c = 0$, we

[1] It is possible that A or C could be composed of a set of parameters.

obtain $\mathcal{P}(A) \propto 1/A$ and $\mathcal{P}(C) \propto 1/C$ and then find that $\ln A$ and $\ln C$ are uniformly distributed. A pleasing consequence of the use of such "improper" hyperpriors is having the property of invariance of inferences. That is, if the parameter A or C is transformed, posterior inference based upon the new parameter will be consistent with those based upon the old parameter (Press, 1988).

3.1.1 Hierarchical Inference

So far, we have set up a general Bayesian framework in weight-space. In the framework, hierarchical Bayesian inference at three different levels can then be carried out step by step (MacKay, 1995).

3.1.1.1 Level 1: Weight Inference

The posterior distribution $\mathcal{P}(\mathbf{w}|\mathcal{D}, C, A)$ (3.3) can also be written as

$$\mathcal{P}(\mathbf{w}|\,\mathcal{D}, C, A) = \frac{1}{\mathcal{Z}_H} \exp\big(-S(\mathbf{w})\big) \tag{3.7}$$

where

$$S(\mathbf{w}) = CE_{\mathcal{D}} + AE_{\mathbf{w}} \tag{3.8}$$

and

$$\mathcal{Z}_H(A, C) = \int \exp\big(-S(\mathbf{w})\big)\, d\mathbf{w} \tag{3.9}$$

Consider first the problem of finding the weight vector $\mathbf{w}_{\mathrm{MP}}$ corresponding to the maximum of the posterior distribution $\mathcal{P}(\mathbf{w}|\mathcal{D}, C, A)$ with fixed hyperparameters. This can be found by minimizing the negative logarithm of (3.7) with respect to the weights. Since the normalizing factor $\mathcal{Z}_H$ in (3.7) is independent of the weights, it is equivalent to minimizing $S(\mathbf{w})$ given by (3.8), i.e. $\mathbf{w}_{\mathrm{MP}} = \arg\min_{\mathbf{w}} S(\mathbf{w})$.

Prior Distribution on Weight It is common to choose zero-mean Gaussian distribution with variance A^{-1} for the prior distribution

$$\mathcal{P}(\mathbf{w}|\,A) = \left(\frac{A}{2\pi}\right)^{\frac{W}{2}} \exp\left(-\frac{A}{2}\|\mathbf{w}\|^2\right) \tag{3.10}$$

where W is the total number of weights (including the bias w_0). This Gaussian distribution expresses our initial knowledge about the weight $\mathbf{w}$ in the data modelling task before any data

are available. When $\|\mathbf{w}\|^2$, i.e. $\mathbf{w}^T\mathbf{w}$, is large, $\mathcal{P}(\mathbf{w}|\,A)$ is small, and so the choice of prior distribution says that we expect the weight values to be small rather than large. Smaller weight values result in a simpler model with smooth output. A major advantage of the choice on the prior distribution (3.10) is that the Gaussian is of nice properties to simplify some of the analysis. Many other choice for the prior can also be considered, such as the Laplacian prior with $E_{\mathbf{w}} = \sum_{i=1}^{W} |w_i|$ (Williams, 1995) and entropy-based priors discussed by Burtine and Weigend (1991). The appropriate selection of priors for very large networks is discussed by Neal (1996).

Likelihood Evaluation We can rewrite the likelihood given the weight $\mathbf{w}$ as

$$\mathcal{P}(\mathcal{D}|\mathbf{w}, C) = \prod_{i=1}^{n} \mathcal{P}(x_i, y_i|\mathbf{w}, C) = \prod_{i=1}^{n} \mathcal{P}(y_i|\mathbf{w}, C, x_i)\mathcal{P}(x_i|\mathbf{w}, C) \tag{3.11}$$

Since the probability of input vector x_i does not depend on the weight vector $\mathbf{w}$ and the hyperparameter C, the conditional likelihood is equal to

$$\mathcal{P}(\mathcal{D}|\mathbf{w}, C) = \prod_{i=1}^{n} \mathcal{P}(y_i|\mathbf{w}, C, x_i)\mathcal{P}(x_i) \tag{3.12}$$

Furthermore, as we have pointed out in Chapter 2, the MLP networks trained in supervised learning do not model the distribution $\mathcal{P}(\boldsymbol{x})$ of the input data at all. As a constant term independent of the weights and the hyperparameters, $\mathcal{P}(x_i)$ in the conditional likelihood (3.12) could be ignored to simplify the notation. In other words, we can always assume that the input vectors $\boldsymbol{x}$ appear as conditioning variables from now on. The other term $\mathcal{P}(y_i|\mathbf{w}, C, x_i)$ in (3.12) represents a model for the noise on the target data as in (2.5), i.e.

$$\mathcal{P}(y_i|\mathbf{w}, C, x_i) = \frac{1}{\mathcal{Z}(C)} \exp\Big(- C \cdot \ell\big(y_i - f(x_i; \mathbf{w})\big)\Big). \tag{3.13}$$

where $\mathcal{Z}(C)$ is a normalizing factor, $\ell(\cdot)$ could be any loss function we have discussed in Chapter 2 and $f(x; \mathbf{w})$ is the output of the MLP model. With a given weight vector $\mathbf{w}$, we can get the model output for input vector x (see Figure 3.2) as

$$f(x; \mathbf{w}) = \sum_{i=1}^{m} w_i \cdot \varphi\left(\sum_{j=1}^{d} w_i^j \cdot x^j\right) + w_0 \tag{3.14}$$

where the actuation function $\varphi(\cdot)$ is usually the logistic (sigmoid) function or the hyperbolic tangent (odd sigmoid) function (Haykin, 1999). Thus, using (3.13), $\mathcal{P}(\mathcal{D}|\mathbf{w}, C)$ can be written

in the form of loss function

$$\mathcal{P}(\mathcal{D}|\mathbf{w}, C) = \left(\frac{1}{\mathcal{Z}(C)}\right)^n \exp\left(-C\sum_{i=1}^{n}\ell\big(y_i - f(x_i;\mathbf{w})\big)\right) \tag{3.15}$$

Discussion For the particular prior distribution given by (3.10) and the quadratic loss function (2.9) in likelihood (3.15), we can get an optimization problem for weight inference as

$$\min_{\mathbf{w}} S(\mathbf{w}) = \frac{C}{2}\sum_{i=1}^{n}\big(y_i - f(x_i;\mathbf{w})\big)^2 + \frac{A}{2}\mathbf{w}^T\mathbf{w} \tag{3.16}$$

We can see that, apart from an overall multiplicative factor, $S(\mathbf{w})$ as in (3.16) is precisely the usual sum-of-square error function with a weight decay regularization term (Hinton, 1987). Note that, in finding the weight vector $\mathbf{w}_{\mathrm{MP}}$, the effective regularization parameter depends only on the ratio A/C, since the overall multiplicative factor is trivial.

Now let us consider a succession of training sets with increasing numbers n of patterns. We can see that the first term in (3.16) or (3.8) increases, while the second term does not. If the hyperparameters A and C are fixed, then as n training data size increases, the first term becomes more and more dominant and the second term becomes eventually insignificant. The maximum likelihood solution is then a very good approximation to the solution $\mathbf{w}_{\mathrm{MP}}$. Conversely, for very small data sets the prior term plays an important role in the solution of $\mathbf{w}_{\mathrm{MP}}$ that prevents the data modelling from over-fitting.

3.1.1.2 Level 2: Hyperparameter Inference

So far, we have assumed that the hyperparameters are fixed parameters with known values in the model $\mathcal{M}_m$. However, these hyperparameters A and C should be regarded as random variables too in Bayesian frameworks. Let us collect C and A as θ, the hyperparameter vector. Once we observe the training data $\mathcal{D}$, we can write down an expression for the posterior probability distribution for the hyperparameter vector θ, which we denote by $\mathcal{P}(\theta|\mathcal{D})$, using Bayes' theorem,

$$\mathcal{P}(\theta|\mathcal{D}) = \frac{\mathcal{P}(\mathcal{D}|\,\theta)\mathcal{P}(\theta)}{\mathcal{P}(\mathcal{D})} \tag{3.17}$$

where the denominator is a normalizing factor can be written as

$$\mathcal{P}(\mathcal{D}) = \int \mathcal{P}(\mathcal{D}|\theta)\mathcal{P}(\theta)\,d\theta \tag{3.18}$$

To fully evaluate the posterior distribution (3.17), the prior probability distribution $\mathcal{P}(\theta)$ is required. The choice on $\mathcal{P}(\theta)$ is quite subjective. The facts we exactly know may be that these hyperparameters are independent and greater than zero. One possible choice is the Gamma distribution as in (3.5) and (3.6). If we have no further idea of what would be suitable values for A and C, then we should choose a prior which in some sense gives equal weight to all possible values. Such priors are called non-informative and are discussed at length in Berger (1985).[2] The maximum a posteriori (MAP) estimate of θ refers to $\arg\max_{\theta} \mathcal{P}(\theta|\mathcal{D})$.

To find A_{MP} and C_{MP} corresponding to the maximum of the posterior distribution, MacKay (1992c) proposed an approach known as evidence approximation. As we typically have little idea about the suitable values of θ before the 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, the likelihood $\mathcal{P}(\mathcal{D}|\,\theta)$ can be used to assign a preference to alternative values of the hyperparameters θ, which is called the evidence of θ. Notice that the normalizing factor $\mathcal{P}(\mathcal{D}|\,\theta)$ in (3.4)[3] is just the likelihood term $\mathcal{P}(\mathcal{D}|\,\theta)$ in (3.17). The evidence (3.4) can be calculated by an explicit formula after using a Laplacian approximation at $\mathbf{w}_{\mathrm{MP}}$. Using (3.1) and (3.2) in (3.4), we can get

$$\mathcal{P}(\mathcal{D}|\theta) = \frac{1}{\mathcal{Z}_{\mathbf{w}}(A)} \frac{1}{\mathcal{Z}_{\mathcal{D}}(C)} \int \exp\big(-S(\mathbf{w})\big)\, d\mathbf{w} \tag{3.19}$$

where $S(\mathbf{w})$ is defined as in (3.8). At the $\mathbf{w}_{\mathrm{MP}}$, a Laplacian approximation on $S(\mathbf{w})$ can be carried out by the Taylor expansion and retaining terms up to second order, i.e.

$$S(\mathbf{w}) \approx S(\mathbf{w}_{\mathrm{MP}}) + \frac{1}{2}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}})^T \cdot \mathbf{H} \cdot (\mathbf{w}-\mathbf{w}_{\mathrm{MP}}) \tag{3.20}$$

where $\mathbf{H} = \nabla\nabla S(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_{\mathrm{MP}}} = A\,\nabla\nabla E_{\mathbf{w}}|_{\mathbf{w}=\mathbf{w}_{\mathrm{MP}}} + B\,\nabla\nabla E_{\mathcal{D}}|_{\mathbf{w}=\mathbf{w}_{\mathrm{MP}}}$. The marginalization in (3.19) can then be analytically calculated as

$$\mathcal{P}(\mathcal{D}|\theta) \approx \frac{\exp\big(-S(\mathbf{w}_{\mathrm{MP}})\big)(2\pi)^{\frac{W}{2}}(\det \mathbf{H})^{-\frac{1}{2}}}{\mathcal{Z}_{\mathbf{w}}(A)\mathcal{Z}_{\mathcal{D}}(C)} \tag{3.21}$$

We have shown that the evidence could quantify Occam's razor automatically in Section 1.2.2. To find the optimal θ_{MP} that maximizes the evidence $\mathcal{P}(\mathcal{D}|\theta)$, we can make use of the gradient information of $\mathcal{P}(\mathcal{D}|\theta)$ (3.21) with respect to θ, and then a standard iterative training algorithm can be used to infer θ_{MP} (MacKay, 1992c; Bishop, 1995). [4]

[2] Non-informative priors for scale parameters, such as A and C, are generally chosen to be uniform on a logarithmic scale.

[3] The model $\mathcal{M}_m$ is an implicit conditional random variable on the right-hand side of the probabilities in (3.3).

[4] Based on the above evidence approximation, it is straightforward to find the MAP estimate, i.e., $\arg\max_{\theta} \mathcal{P}(\mathcal{D}|\theta)\mathcal{P}(\theta)$, where $\mathcal{P}(\mathcal{D}|\theta)$ is approximated by (3.21).

There is a potential difficulty in the evidence approximation. For a general non-linear network mapping function $f(x; \mathbf{w})$ as in (3.14), there may be numerous local minima of the error function $E_{\mathcal{D}}$, some of which may be associated with symmetries in the network (Chen et al., 1993). The single-Gaussian approximation at one local minimum given by (3.21) clearly does not take multiple minima into account. Thus maximization of the evidence quantity could be a poor solution if the Laplacian approximation failed to give a good summary of the distribution.

3.1.1.3 Level 3: Model Comparison

Recall that we have prepared a set of models with different hidden neurons $\{\mathcal{M}_m\}$ for the data modelling task. To rank alternative models in the light of the training data $\mathcal{D}$, we examine the posterior probabilities of alternative models $\mathcal{M}_m$:

$$\mathcal{P}(\mathcal{M}_m|\mathcal{D}) \propto \mathcal{P}(\mathcal{D}|\mathcal{M}_m)\mathcal{P}(\mathcal{M}_m) \tag{3.22}$$

The data-dependent term, the evidence of $\mathcal{M}_m$, appeared earlier as the normalizing factor in (3.18).[5] Assuming that we have no reason to assign strongly differing priors $\mathcal{P}(\mathcal{M}_m)$, alternative models $\mathcal{M}_m$ are ranked just by examining the evidence. The model evidence $\mathcal{P}(\mathcal{D}|\mathcal{M}_m)$ is exactly an integral over weight space and hyperparameter space

$$\mathcal{P}(\mathcal{D}|\mathcal{M}_m) = \int\int \mathcal{P}(\mathcal{D}|\mathbf{w}, C)\mathcal{P}(\mathbf{w}|A)\mathcal{P}(\theta)\, d\mathbf{w}\, d\theta \tag{3.23}$$

that can not be calculated analytically. Some approximation approaches have been discussed in MacKay (1992c); Bishop (1995). Hybrid Mente Carlo methods (Duane et al., 1987; Neal, 1996) could also be applied to approximate the integral.

We have introduced the classical three levels of inference in the Bayesian framework. The Bayesian techniques for neural networks specify a hierarchical model with a prior distribution over hyperparameters, and then a prior distribution for the weights governed by the hyperparameters. This induces a posterior distribution over the weight and the hyperparameters for a given data set. We can simply pick up the most probable points in these posterior distributions as the optimal solution, or integrate over these distributions for higher level inference or the future predictions.

[5] The model $\mathcal{M}_m$ is an implicit conditioning variable on the right-hand side of the distributions in (3.18).

3.1.2 Distribution of Network Outputs

Suppose that we have determined the most plausible model in the model set $\{\mathcal{M}_m\}$, and then found the most probable hyperparameter vector θ_{MP} in the posterior distribution (3.17). As we have seen, in the Bayesian formalism a "trained" network is described in terms of the posterior probability distribution of weight values. If we present a new input vector x to such a network, then the distribution of weights gives rise to a distribution of network outputs. In addition, there will be a contribution to the output distribution arising from the assumed noise on the output variables.

Using a selected network $\mathcal{M}_m$ with the given hyperparameter vector θ, we can write the distribution of outputs, for a given input vector x, in the form

$$\mathcal{P}(y|x,\theta,\mathcal{M}_m,\mathcal{D}) = \int \mathcal{P}(y|x,\mathbf{w},\theta,\mathcal{M}_m,\mathcal{D})\mathcal{P}(\mathbf{w}|\theta,\mathcal{M}_m,\mathcal{D})\,d\mathbf{w} \tag{3.24}$$

where $\mathcal{P}(\mathbf{w}|\theta,\mathcal{M}_m,\mathcal{D})$ is the posterior distribution of weights evaluated by (3.7), and the distribution $\mathcal{P}(y|x,\mathbf{w},\theta,\mathcal{M}_m,\mathcal{D})$ is the model for the additive noise on the target given in (3.13).

If we choose the quadratic loss function (2.9) in noise model, then a Gaussian distribution could be used to approximately evaluate (3.24). Using the Laplacian approximation of $S(\mathbf{w})$ at $\mathbf{w}_{\mathrm{MP}}$ as in (3.20), there is a single-Gaussian approximation for (3.7)

$$\mathcal{P}(\mathbf{w}|\theta,\mathcal{M}_m,\mathcal{D}) \propto \exp\left(-\frac{1}{2}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}})^T\cdot\mathbf{H}\cdot(\mathbf{w}-\mathbf{w}_{\mathrm{MP}})\right) \tag{3.25}$$

With the quadratic loss function (2.9) and the single-Gaussian approximation (3.25), (3.24) can be written as

$$\mathcal{P}(y|x,\theta,\mathcal{M}_m,\mathcal{D}) \propto \int \exp\left(-\frac{C}{2}\big(y-f(x;\mathbf{w})\big)^2-\frac{1}{2}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}})^T\cdot\mathbf{H}\cdot(\mathbf{w}-\mathbf{w}_{\mathrm{MP}})\right)d\mathbf{w} \tag{3.26}$$

where we have dropped any factors independent of y. In addition, we shall assume that the width of the posterior distribution $\mathcal{P}(\mathbf{w}|\theta,\mathcal{M}_m,\mathcal{D})$ is sufficiently narrow that we may approximate the network function $f(x;\mathbf{w})$ by its linear expansion around $\mathbf{w}_{\mathrm{MP}}$

$$f(x;\mathbf{w}) \approx f(x;\mathbf{w}_{\mathrm{MP}}) + \mathbf{g}^T\triangle\mathbf{w} \tag{3.27}$$

where $\mathbf{g} = \nabla_{\mathbf{w}} f(x;\mathbf{w})|_{\mathbf{w}=\mathbf{w}_{\mathrm{MP}}}$. Introducing (3.27) into (3.26), the integral is easily evaluated to

give a Gaussian distribution of the form (Bishop, 1995)

$$\mathcal{P}(y|x,\theta,\mathcal{M}_m,\mathcal{D}) = \frac{1}{(2\pi\sigma_y^2)^{1/2}}\exp\left(-\frac{\big(y - f(x;\mathbf{w}_{\mathrm{MP}})\big)^2}{2\sigma_y^2}\right) \tag{3.28}$$

where we have restored the normalization factor explicitly and the variance of the Gaussian distribution is given by

$$\sigma_y^2 = \frac{1}{C} + \mathbf{g}^T\mathbf{H}^{-1}\mathbf{g} \tag{3.29}$$

We see that the Bayesian formalism allows us to calculate error bars (3.29) on the network outputs, instead of just providing a single best guess output as other deterministic approaches do. The error bar in (3.29) has two contributions: one comes from the intrinsic noise on the target data, corresponding to the first term in (3.29),[6] and another arising from the width of the posterior distribution of the network weights, corresponding to the second term in (3.29).

We can also integrate the predictive distribution over the posterior distribution $\mathcal{P}(\theta|\mathcal{M}_m,\mathcal{D})$ to make predictions

$$\mathcal{P}(y|x,\mathcal{M}_m,\mathcal{D}) = \int \mathcal{P}(y|x,\theta,\mathcal{M}_m,\mathcal{D})\mathcal{P}(\theta|\mathcal{M}_m,\mathcal{D})\,d\theta \tag{3.30}$$

where $\mathcal{P}(\theta|\mathcal{M}_m,\mathcal{D})$ is the posterior distribution of the hyperparameters given by (3.17) and $\mathcal{P}(y|x,\theta,\mathcal{M}_m,\mathcal{D})$ is given by (3.24). Comparing with the predictive distribution (3.24) we made using the optimal hyperparameter vector θ_{MP}, the integration over hyperparameter space could erase the uncertainty in θ by taking all the possible choices of θ into account. However, the integral can not be done analytically, Monte Carlo methods (Duane et al., 1987; Neal, 1996) could be used here to approximate the marginalization.

In a full Bayesian treatment, we have to go further to integrate over the model set $\{\mathcal{M}_m\}$, i.e.

$$\mathcal{P}(y|x,\mathcal{D}) = \sum_m \mathcal{P}(y|x,\mathcal{M}_m,\mathcal{D})\mathcal{P}(\mathcal{M}_m|\mathcal{D}) \tag{3.31}$$

where $\mathcal{P}(\mathcal{M}_m|\mathcal{D})$ is the posterior probability of the model $\mathcal{M}_m$ given by (3.22) and $\mathcal{P}(y|x,\mathcal{M}_m,\mathcal{D})$ is given by (3.30), and then even further consider the integral

$$\mathcal{P}(y|x) = \int \mathcal{P}(y|x,\mathcal{D})\mathcal{P}(\mathcal{D})\,d\mathcal{D} \tag{3.32}$$

to erase the randomness in the collection of training data.

[6] The first term in (3.29), $\frac{1}{C}$, is the variance of the Gaussian noise.

It should be noted that computational cost could be very expensive to evaluate these marginalizations in the strict Bayesian formalism. For a particular learning task, we need to infuse our prior knowledge into the formalism effectively and then try to simplify the problem at hand by reasonable approximations.

3.1.3 Some Variants

As a very flexible and powerful framework, standard Bayesian formalism could bring forth lots of useful variants. For a particular learning task, it is common that some features are more important than others, or some samples are more useful than others in the prediction of the target. We now discuss variants of these Bayesian frameworks that can incorporate these properties.

3.1.3.1 Automatic Relevance Determination

In many problems, there is a large number of potential measurable features (or attributes) that could be included as inputs. However, including more input features must ultimately lead to poor performance due to over-fitting. The subsequent predictive performance on unseen test data will then be poor. Accordingly, we need to assess the relevance level of each feature.

If we do include many input features that we think are probably irrelevant, we would like to use models that can automatically determine the degree of the relevance of each input feature. Such a model has been developed by MacKay (1995) and Neal (1996), which is known as *Automatic Relevance Determination* (ARD) model. In an ARD model, each input variable is associated with an ARD hyperparameter that controls the magnitudes of the weights on connections of that input feature. These hyperparameters are given some prior distribution, and then conditional on the values of these ARD hyperparameters, the weights connected with each input feature have been specified as independent zero-mean Gaussian prior distributions with standard deviation given by the corresponding ARD hyperparameter values. If the ARD hyperparameter associated with a feature specifies a small standard deviation for weights connected with that feature, all of these weights will likely be very small, and then the feature will have little effect on the model output; if the ARD hyperparameter specifies a large standard deviation, the weights could likely be large values and then the effect of the feature in the output will be significant. The posterior distributions of these ARD hyperparameters will show which hyperparameters are most plausible in the light of the training data.

We now choose the MLP networks in Figure 3.2 to illustrate how to set up the ARD model. Let us categorize the weights into $d + 1$ groups, i.e. the weights associated the input features

x^j as an weight vector $\mathbf{w}^j = [w_1^j, w_2^j, \ldots, w_m^j]^T$ where m is the number of hidden neurons and $j = 1, 2, \ldots, d$, and the other weights as $\mathbf{w}^0 = [w_0, w_1, \ldots, w_m]^T$.[7] We then specify an independent Gaussian distribution for each weight vector $\mathbf{w}^i$

$$\mathcal{P}(\mathbf{w}^i|\kappa_i) = \sqrt{\frac{\kappa_i}{2\pi}} \exp\left(-\frac{\kappa_i}{2}\|\mathbf{w}^i\|^2\right) \tag{3.33}$$

where $i = 0, 1, \ldots, d$ and $\kappa_i > 0$. Here κ_i is known as the ARD hyperparameter for the i-th weight class which controls the shape of the Gaussian distribution (3.33). When κ_i is of a small value, the standard deviation of the Gaussian is then a large value and that means the Gaussian is distributed broadly. Thus the weights could likely be realized as large values that make the effect of the weight class in model output significant. On the contrary, for a narrowly distributed zero-mean Gaussian, the weights are very likely to be small values around zero that produce trivial effect on the model. Therefore, the values of these hyperparameters associated weight groups could be regarded as a measure on the relevance of the corresponding features.

The joint distribution of these weight groups $\mathbf{w}^i$ is still a Gaussian that is

$$\mathcal{P}(\mathbf{w}|\kappa_i) = \sqrt{\frac{\prod_{i=0}^{d} \kappa_i}{(2\pi)^{d+1}}} \exp\left(-\sum_{i=0}^{d} \frac{\kappa_i}{2}\|\mathbf{w}^i\|^2\right) \tag{3.34}$$

due to the independence. Notice that (3.34) is just a special case of the general prior distribution (3.1) in the weight-space. Thus it is straightforward to apply hyperparameter inference and other higher level inference on the ARD model.

3.1.3.2 Relevance Vector Machines

The ARD model could infer the relevance level of the features of the training samples in the multilayer perceptron networks as we have discussed, while relevance vector machines (Tipping, 2000) could infer the relevance level of the basis (or kernel) functions in fixed basis function networks. In the following, we shall give a brief review on this interesting formulation.

Let us first consider the Bayesian approach to fixed basis function networks. The fixed basis function model is usually defined as

$$f(x; \mathbf{w}) = \sum_{i=1}^{m} w_i \cdot \Phi_i(x) + w_0, \tag{3.35}$$

[7] Of course, we might separate this class $\mathbf{w}^0$ into more groups. For instance, we could separate the bias and the weights out of hidden neurons as two different groups. We can even set each of the weights out of hidden neurons as individual class.

where $\{\Phi_i\}$ is a set of basis functions. If the fixed basis functions $\{\Phi_i\}$ are chosen as a set of radial-basis functions $\{\Phi_i(\|x - u_i\|)|i = 1, 2, \ldots, m\}$ where $\|\cdot\|$ denotes a norm that is usually Euclidean and u_i is the center, then (3.35) becomes the model given by RBF networks.

Let us collect the weight vector $\mathbf{w} = [w_0, w_1, \ldots, w_m]^T$, and specify a prior (3.1) on the weights $\mathbf{w}$ such as

$$\mathcal{P}(\mathbf{w}|A) = \frac{1}{\mathcal{Z}_{\mathbf{w}}(A)} \exp\left(-\frac{A}{2}\mathbf{w}^T\mathbf{w}\right), \tag{3.36}$$

and then choose a noise model in likelihood function, usually a Gaussian noise model

$$\mathcal{P}(y_i|x_i, \mathbf{w}, C) = \frac{1}{\mathcal{Z}(C)} \exp\left(-\frac{C}{2}\big(y_i - f(x_i; \mathbf{w})\big)^2\right) \tag{3.37}$$

where i takes values from 1 to n, and A and C are appropriate hyperparameters. By integrating over the weight space, we can show the joint probability of the targets given the inputs as

$$\mathcal{P}(\boldsymbol{y}|\boldsymbol{x}, A, C) = \frac{(\det \mathbf{G})^{\frac{1}{2}}}{(2\pi)^{\frac{n}{2}}} \exp\left(-\frac{1}{2}\boldsymbol{y}^T\mathbf{G}\,\boldsymbol{y}\right) \tag{3.38}$$

where $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^T$ and $\mathbf{G} = \frac{1}{A}\mathbf{\Phi}\mathbf{\Phi}^T + \frac{1}{C}\mathbf{I}$ with $\mathbf{\Phi}_{ij} = \Phi_j(x_i)$. Comparing (3.38) with the normalization factor (3.4), we can see that the joint probability of the data is just the evidence of these hyperparameters. A gradient based optimization algorithm can then be used to find the most probable hyperparameters or Monte Carlo sampling methods (such as Hybrid Monte Carlo) can be used to integral over the hyperparameters approximately.

Relevance vector machine (RVM) proposed by Tipping (2000) is a general Bayesian framework for the basis function model to obtain sparse solutions. Tipping (2001) focused on a particular specialization that is a model of identical functional form to the popular SVMs, i.e. $m = n$ and the basis function $\Phi_i(x) = \phi(x, x_i)$ which could be any form even other than kernel function $K(x, x_i)$. In this approach, a zero-mean Gaussian prior distribution is assigned to each weight w_i as follows:

$$\mathcal{P}(\mathbf{w}|\mathcal{A}) = \prod_{i=0}^{n} \mathcal{P}(w_i|A_i) = \prod_{i=0}^{n} \sqrt{\frac{A_i}{2\pi}} \exp\left(-\frac{A_i}{2}w_i^2\right), \tag{3.39}$$

with $n + 1$ hyperparameters $\mathcal{A} = \{A_i|i = 0, 1, \ldots, n\}$. This is just a joint $n + 1$ multivariate Gaussian

$$\mathcal{P}(\mathbf{w}|\mathcal{A}) = \frac{|\mathbf{A}|^{\frac{1}{2}}}{(2\pi)^{\frac{n+1}{2}}} \exp\left(-\frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w}\right), \tag{3.40}$$

with the covariance matrix $\mathbf{A} = \mathrm{diag}(A_0, A_i, \ldots, A_n)$ where one hyperparameter associated with

each weight to control the prior distribution over the weights. Same as the principle of ARD models, the value of the hyperparameter A_i determines the significance of the associated basis function $\Phi_i(x)$ in the output of the model. A large value of A_i means the weight w_i connected with $\Phi_i(x)$ is narrowly distributed around zero and then the value of w_i is quite likely to be small value that makes the effect of $\Phi_i(x)$ in the output (3.35) trivial. While a smaller A_i makes the weight w_i widely distributed that implies more importance of $\Phi_i(x)$ in the model output.

In the case that Gaussian noise model (3.37) is used, the evidence of these hyperparameters (3.38) can then be evaluated with $\mathbf{G} = \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T + \frac{1}{C}\mathbf{I}$ and $\mathbf{\Phi}_{ij} = \Phi(x_i, x_j)$. For the case of the uniform hyperpriors, i.e. uniform distribution on the logarithmic scale, we need only maximize the evidence of these hyperparameters, which is known as the type-II maximum likelihood method (Berger, 1985). The most probable values of the set of hyperparameters will be iteratively estimated from the data (Tipping, 2001).

The most compelling feature of the RVM is that it typically utilizes dramatically few kernel functions. The sparseness is achieved because Tipping (2001) reported that in practice the posterior distributions of many of the weights are sharply peaked around zero, i.e. the associated hyperparameter tends to be infinite in the evidence maximization. A threshold is used to prune these kernel functions associated with "infinite" hyperparameters.[8] Due to the sparsity in the solution, RVM is regarded as a principled Bayesian alternative to SVMs (Schölkopf and Smola, 2001).

3.2 Gaussian Processes

The study of Gaussian processes for regression is far from new. The idea has been used for a long time in the spatial statistics community under the name "kriging", see Cressie (1993) for a review, although it seems to have been concentrated mainly on low-dimensional input space and largely ignored any probabilistic interpretation of the model. Williams and Rasmussen (1996) extended the use of Gaussian process prior to higher dimensional regression problems and good results have been obtained. Regression with Gaussian processes (GPR) is reviewed by Williams (1998).

The Gaussian process framework encompasses a wide range of different regression models. O'Hagan (1978) introduced an approach which is essentially similar to Gaussian processes which are widely used in the analysis of computer experiments, although in this application it is assumed

[8]The threshold is crucial since it potentially determines the structure of the network and its generalization.

that the observations are noise free. A connection to neural networks was made by Poggio and Girosi (1990) with their work on regularization networks. When the covariance function depends only on $\|x_i - x_j\|$, the predictor derived by Gaussian processes might be same as that made by generalized radial basis function (or RBF) networks. Wahba (1990) provides a useful overview on the use of spline techniques for regression problems. Her work dated back to Kimeldorf and Wahba (1971). Essentially splines correspond to Gaussian processes with a particular choice of covariance function. The variable metric kernel methods (Lowe, 1995) is also closely related to Gaussian processes. A comparison of Gaussian processes with other methods such as neural networks has been done by Rasmussen (1996).

3.2.1 Covariance Functions

Formally, a Gaussian process is a collection of random variables $\{f(x)\}$ indexed by a set of $x \in \mathbb{R}^d$, where any finite subset of these random variables has a joint Gaussian distribution. The Gaussian process is fully specified by its mean and its covariance function, $Cov[f(x), f(x')]$. The covariance function is defined as $Cov[f(x), f(x')] = E\big[\big(f(x) - E[f(x)]\big)\big(f(x') - E[f(x')]\big)\big]$ where $E(f(x))$ is the mean of the random variable $f(x)$, which is only a function of the inputs x and x', i.e. $Cov(x, x')$. We follow the classical settings to specify the mean as zero thereafter for simplicity. Then the covariance becomes $E[f(x)f(x')]$ which is also known as autocorrelation, and the matrix of covariances between pairs of input patterns is referred to as covariance matrix.

As we have mentioned before, the covariance function and therefore the covariance matrix plays a pivotal role in the Gaussian process model. The predictive distributions that are derived for given data sets are highly dependent on the covariance function and its hyperparameters. There are some constraints on the form of the covariance function. Formally, we are required to specify a function which will generate a positive definite covariance matrix for any set of distinct input patterns in order to ensure that the distribution is normalizable. We also wish to express our prior beliefs about the modelling task, i.e. the structure of the underlying function. In the following, we shall discuss various choices for the covariance function and their parameterization.

3.2.1.1 Stationary Components

An stationary covariance function dependents on the relative position of the two input patterns, that is satisfying $Cov(x, x + h; \theta) = Cov(h; \theta)$ for one dimensional case and

$$Cov(x, x'; \theta) = Cov(\|x - x'\|; \theta) \tag{3.41}$$

for multidimensional case where θ denotes the vector of the hyperparameters in the covariance function.

In the light of Bochner's theorem (Bochner, 1979), a stationary covariance function $Cov(x - x'; \theta)$, that holds

$$Cov(x - x'; \theta) = \int_{-\infty}^{\infty} \exp\big(\mathtt{i}\,\omega(x - x')\big)\upsilon(\omega)d\omega \tag{3.42}$$

for a positive symmetric measure $\upsilon(\omega)$, must satisfy the positive definite constraint.[9] Clearly, such a covariance function (3.42) is also a Green's function. The equation (3.42) also provides us with an representation of the covariance function in the frequency domain. In the following, we will introduce a wide range of popular kernels that are used not only in support vector machines but also in covariance functions, along with their frequency representations.

Gaussian Kernels Gaussian radial basis function kernels are widely used in neural networks (Haykin, 1999) and support vector machines (Vapnik, 1998), which are defined as

$$K(x - x'; \sigma) = \exp\left(-\frac{(x - x')^2}{2\sigma^2}\right). \tag{3.43}$$

For a Fourier representation we need only to compute the Fourier transform of (3.43), which is given for one dimensional case as

$$\tilde{K}(\omega) = \upsilon(\omega) = |\sigma| \exp\left(-\frac{\sigma^2\omega^2}{2}\right) \tag{3.44}$$

A profile of Gaussian kernel and its Fourier transform is presented in Figure 3.3. From the representation in frequency domain (3.44) and the right graph in Figure 3.3, we find that the contribution of high frequency components in estimates is relatively small, since $\upsilon(\omega)$ decays extremely rapidly. The hyperparameter σ plays the important role to control the band-width in the low-pass process.

The multidimensional case is completely analogous, since it can be decomposed into a product of one-dimensional Gaussians (3.43). The Gaussian kernel for multidimensional cases, i.e. $x \in \mathbb{R}^d$, is defined as

$$K(x - x'; \sigma) = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) = \exp\left(-\frac{1}{2\sigma^2}\sum_{\iota=1}^{d}(x^{\iota} - x'^{\iota})^2\right). \tag{3.45}$$

[9] Here, we consider $x \in \mathbb{R}$ to avoid tedious notation. The result for higher dimensional cases could also be obtained, for instance by integrating over the individual dimensions.

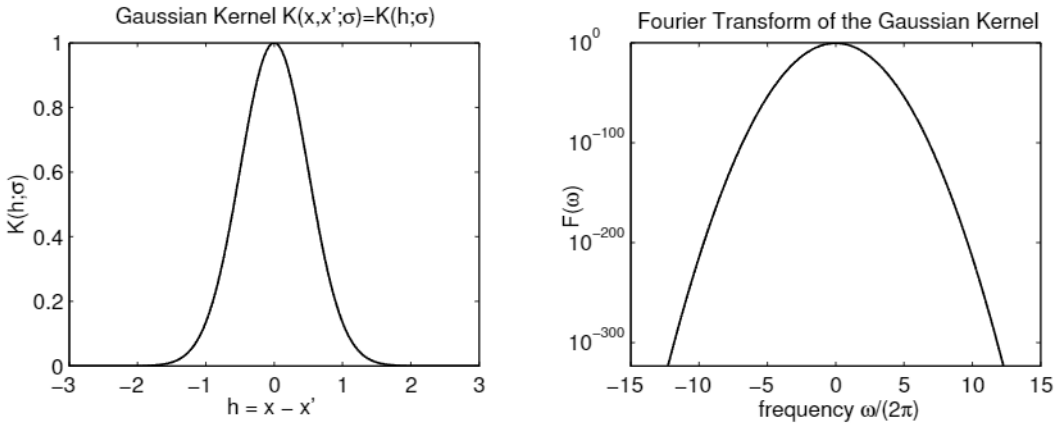

Figure 3.3: Gaussian kernel and its Fourier transform, in which $\sigma = 0.5$.

ARD Gaussian Kernel It is an interesting application of the ARD idea (MacKay, 1995; Neal, 1996) to construct ARD Gaussian kernel by enhancing the Gaussian kernel for multidimensional cases as

$$K(x - x'; \boldsymbol{\sigma}) = \exp\left(-\frac{1}{2}\sum_{\iota=1}^{d}\frac{(x^{\iota} - x'^{\iota})^2}{\sigma_{\iota}^2}\right), \tag{3.46}$$

where the set of hyperparameters $\boldsymbol{\sigma}$ controls the relevance of each dimension to the target. We see that, for the case of $\sigma_{\iota}^2 \to \infty$, the $\iota-$th input dimension is neglected in the computation of the kernel and can therefore be removed in the modelling. The optimal values of $\boldsymbol{\sigma}$ could be found later in Bayesian designs.

B_n-Spline of Odd Order Splines are an important tool in interpolation and function approximation (Wahba, 1990). B_n-Splines was used as kernels in Smola (1996), which is defined as $n+1$ fold convolutions[10] of the centered unit interval $[-0.5, 0.5]$

$$K(x - x'; p) = B_{2p+1}(x) = \bigotimes_{i=1}^{2p+2} I_{[-0.5,0.5]}. \tag{3.47}$$

Given the B_{2p+1}-Spline kernel, we use (3.42) in order to obtain the corresponding Fourier representation. It is well known that convolutions in the original space become products in the Fourier domain and vice versa. Thus the Fourier representation is conveniently given by the $n+1$-th power of the Fourier transform of B_0. Since $F[B_0](\omega) = \mathrm{sinc}(\frac{\omega}{2})$ where $\mathrm{sinc}(x) = \frac{\sin x}{x}$,

[10] A convolution $f \bigotimes g$ of two function $f, g : \mathcal{X} \to \mathbb{R}$ is defined as $f \bigotimes g = \int_{\mathcal{X}} f(t)g(x-t)dt$. Note that $f \bigotimes g = g \bigotimes f$, as can be seen by exchange of variables.

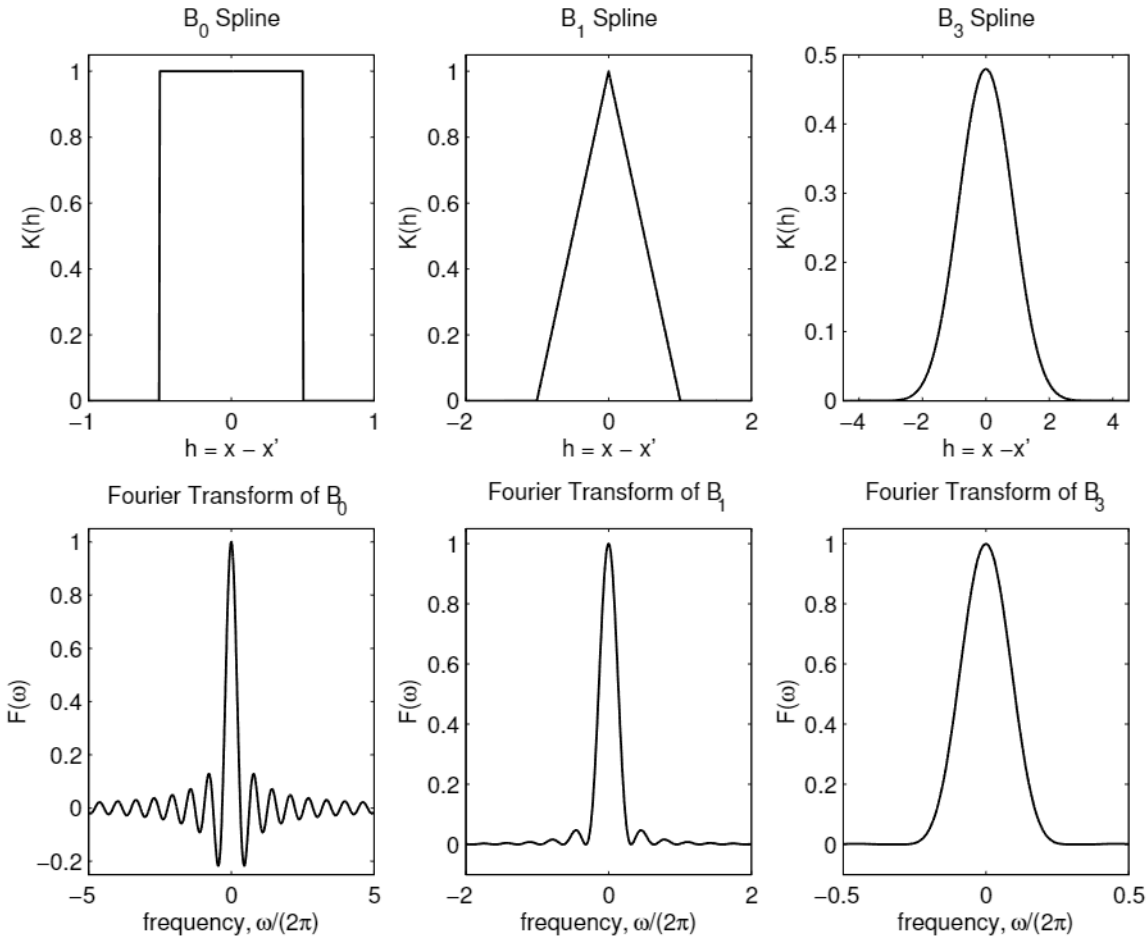

Figure 3.4: Spline kernels and their Fourier transforms. Note that the higher the degree of B_{2p+1}, the more peaked the Fourier transform (3.48) becomes.

we obtain

$$\tilde{B}_{2p+1}(\omega) = v(\omega) = \mathrm{sinc}^{2p+2}\left(\frac{\omega}{2}\right) \tag{3.48}$$

A profile of the one-dimensional case is presented in Figure 3.4.[11] This illustrates why only B_n splines of odd order are positive definite kernels, since the even ones have negative components in the Fourier spectrum that would result in an amplification of the corresponding frequencies (see the left-bottom graph in Figure 3.4).

Dirichlet Kernels The statement of Bochner's theorem (3.42) could also be used to generate practical covariance functions. As a particular choice of $v(\omega)$, Vapnik et al. (1997) used the

[11] The result for higher dimensional cases could also be obtained, for instance by taking products over the individual dimensions.

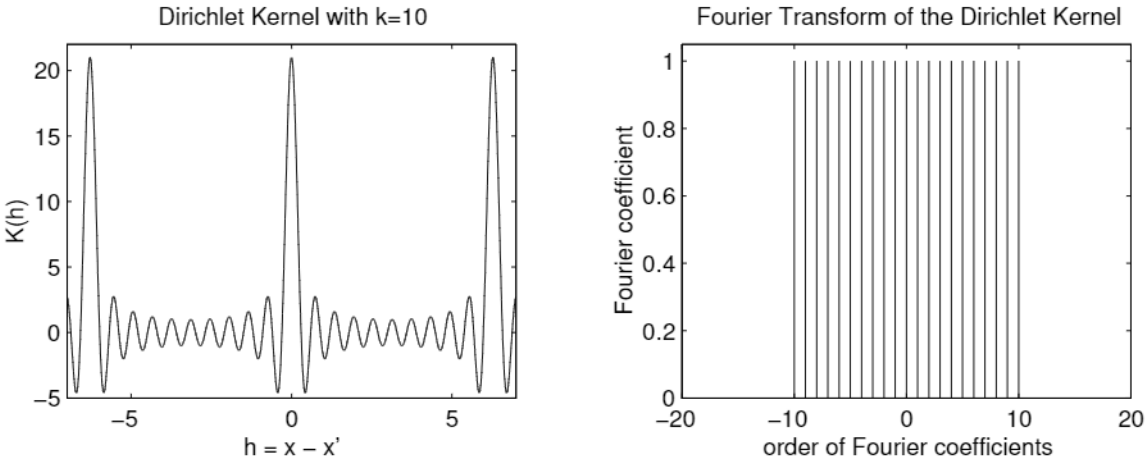

Figure 3.5: Dirichlet kernel with order 10 and its Fourier transform.

Fourier expansions

$$v(\omega) = \sum_{i=-n}^{k} \delta(\omega - i), \tag{3.49}$$

with δ being Dirac's delta function to construct a class of kernel

$$K(x - x'; k) = 2\sum_{j=0}^{k} \cos jx - 1 = \frac{\sin(2n+1)\frac{x}{2}}{\sin\frac{x}{2}}. \tag{3.50}$$

A profile is given in Figure 3.5. This kernel only describes band-limited periodic functions where no distinction is made between the different components of the frequency spectrum. In some cases, it might be useful to approximate periodic functions, for instance functions defined on a circle (Schölkopf et al., 2001).

Note that we could design more covariance functions with specific properties we desire, such as translation invariance, periodicity etc, since this is convenient way of building covariance function if the Fourier expansion $v(\omega)$ is known.

3.2.1.2 Non-stationary Components

While many data sets can be effectively modelled using a stationary covariance function, there are some cases in which we would like some of non-stationarity in our covariance function.

Linear Component Sometimes, we may believe that there is some linear trend in the data. Consider a plane $f(x) = \sum_{\iota=1}^{d} a_\iota x^\iota + b$, in which the $\{a_\iota\}$ and b have Gaussian distribution with

zero mean and variance σ_a^2 and σ_b^2 respectively. The plane then has a covariance function

$$Cov(x, x'; \sigma_a, \sigma_b) = E\big(f(x), f(x')\big) = \sigma_a^2 \sum_{\iota=1}^{d} x^\iota x'^\iota + \sigma_b^2 \tag{3.51}$$

Notice that the term $\sum_{\iota=1}^{d} x^\iota x'^\iota$ is just the linear kernel $K(x, x') = \langle x \cdot x' \rangle$ used in support vector machines. Using the linear covariance function (3.51) only in modelling will yield a linear plane; adding (3.51) into the stationary covariance functions will produce a linear component to the predictions.

We can further assume that the parameters $\{a_\iota\}$ for each dimension have different variance ${\sigma_a^\iota}^2$ instead of the common variance σ_a^2, and then the ARD linear covariance function is obtained as

$$Cov(x, x'; \boldsymbol{\sigma}_a, \sigma_b) = \sum_{\iota=1}^{d} {\sigma_a^\iota}^2 x^\iota x'^\iota + \sigma_b^2 \tag{3.52}$$

where the ARD hyperparameters $\{{\sigma_a^\iota}^2\}$ determine the relevance of each dimension to the target.

Beyond the popular linear component, we can insert other kinds of non-stationary components into the covariance function. For instance, we may get some prior knowledge that the noise level is varying as we cross the input space. Some input-dependent terms could be introduced into the diagonal elements of the covariance matrix to express the noise dependency (Gibbs, 1997).

3.2.1.3 Generating Covariance Functions

So far we have listed some widely used covariance functions which could also work as kernels in support vector methods (or other kernel methods). The sum of any positive definite functions is also positive definite and the same is also true of the product of two positive definite functions. Hence we can generate new covariance functions using simpler covariance functions as the building blocks. For example, it is common to use the sum of ARD Gaussian kernel (3.46) and linear kernel (3.51) as covariance function in standard Gaussian processes for regression (Williams and Rasmussen, 1996; Rasmussen, 1996), which is

$$Cov(x, x'; \theta) = \kappa_0 \exp\left(-\frac{1}{2}\sum_{\iota=1}^{d} \kappa_\iota (x^\iota - x'^\iota)^2\right) + \kappa_a \sum_{\iota=1}^{d} x^\iota x'^\iota + \kappa_b \tag{3.53}$$

where the hyperparameter vector $\theta = [\kappa_0, \kappa_1, \ldots, \kappa_d, \kappa_a, \kappa_b]^T$.

The design on the covariance function is pivotal in the techniques of Gaussian processes, since the covariance function completely determines the prior distribution of the functions $\{f(x_i)\}$. In

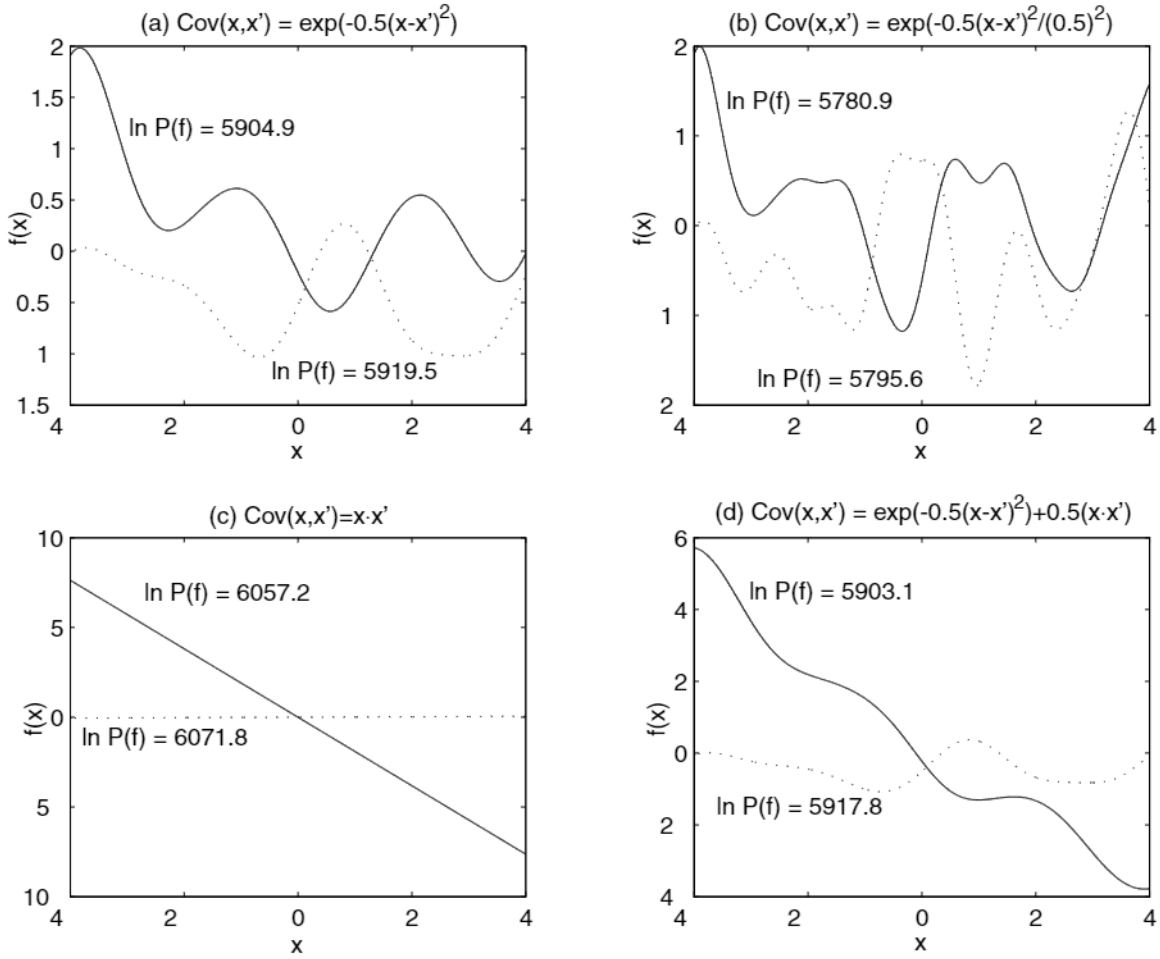

Figure 3.6: Samples drawn from Gaussian process priors. This figure shows two functions drawn from each of four Gaussian process priors with different covariance functions given as titles of each of these graphs. The corresponding prior probabilities of the function values are given by $-\ln \mathcal{P}(f)$ in graphs. The smaller $-\ln \mathcal{P}(f)$ is, the more plausible the model seems.

the following, we show the samples of $\{f(x_i)\}$ from some Gaussian process priors to study the properties of covariance functions.

In Figure 3.6, we notice that the decrease in the variance of Gaussian kernel κ_ι from 1.0 in (a) to 0.25 in (b) produces more rapidly fluctuating function curves. That could also be explained in frequency domain as the expansion of the band-width of the local low-pass filter (3.44). The linear model is seen as straight lines in (c). The covariance function in (d) contains the linear kernel that yields a linear trend in function curves. In Figure 3.7, we present the samples of function values in two-dimensional input space according to the covariance functions given in titles. We notice that the increase in the variance of Gaussian kernel for x^2 from 2.0 in (a) to 10.0 in (b) yields function plane which is quite flat along the axis of x^2, i.e., insensitive to the change of x^2. Comparing the joint probabilities of these samples, we can find that the sample

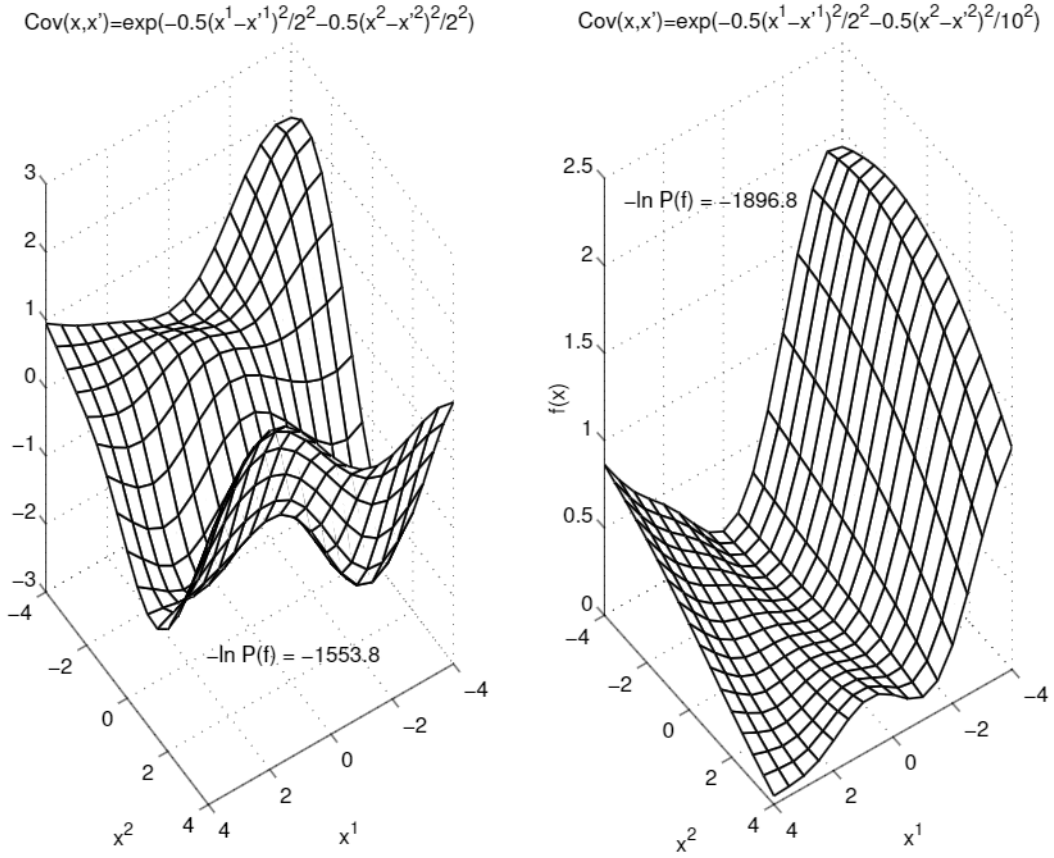

Figure 3.7: Samples drawn from Gaussian process priors in two-dimensional input space. This figure shows the functions drawn from each of two Gaussian process priors with different covariance functions given as titles of the graphs. The corresponding prior probabilities of the function values are given by $-\ln \mathcal{P}(f)$ in graphs.

which represents flatter curves tends to have a higher joint probability. Thus, the Gaussian processes prefer flat curves, i.e. simple models, while punish complex models with lower prior probabilities.

In the one-dimensional case, the input x was 801 linearly equally spaced points between -4 and $+4$. The covariance matrix Σ is a symmetric $n \times n$ matrix whose ij-th element is $Cov(x_i, x_j)$,[12] where n is the size of input data 801. We randomly drew 801 samples in Gaussian distribution $\mathcal{N}(0, 1)$ into a column vector z (We did the sampling twice and kept them same for the four different covariance functions). The function values for each of the four covariance functions are obtained by using affine transformation $f(x) = A \cdot z$ where $\Sigma = A \cdot A^T$ via Cholesky factorization. In the two-dimensional case, the input x is the linearly equally spaced 17×17 grid

[12]Usually a "jitter" term is added in the diagonal entries of the covariance matrix, that contributes additively to every eigenvalues of the matrix and then reducing the condition number. The "jitter" terms could be fixed at the square root of floating point relative accuracy, i.e. the the distance from 1.0 to the next largest floating point number on your system.

covering $[-4,+4] \times [-4,+4]$, other things are similar as that in the one-dimensional case.

3.2.2 Posterior Distribution

In previous section, we have studied some components popularly used in covariance functions of Gaussian processes, and shown the sampling results in the prior distribution defined by various covariance functions. The prior distribution of Gaussian processes is completely specified by its mean and its covariance function. Suppose that we are given a set of data $\mathcal{D}$ composed of n pairs $\{(x_i, y_i)\}_{i=1}^n$. For a given covariance function, the prior distribution of the function values $\boldsymbol{f} = [f(x_1), f(x_2), \ldots, f(x_n)]^T$ is defined as

$$\mathcal{P}(\boldsymbol{f}) = \frac{1}{(2\pi)^{\frac{n}{2}} (\det \Sigma)^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{f} - \boldsymbol{f}_0)^T \cdot \Sigma^{-1} \cdot (\boldsymbol{f} - \boldsymbol{f}_0)\right) \tag{3.54}$$

where $\boldsymbol{f}_0 = E[\boldsymbol{f}]_0 = \left[f_0(x_1), f_0(x_2), \ldots, f_0(x_n)\right]^T$ is the prior mean (usually zero), and the matrix Σ with ij-th entry $Cov(x_i, x_j)$. The subscript 0 denotes the mean is with respect to the prior distribution $\mathcal{P}(\boldsymbol{f})$, i.e. when the GP has not see any data pair. Together with the observed data, the prior distribution of the function values $\boldsymbol{f}$ are converted to posterior distribution through the use of Bayes' theorem. The posterior distribution is the result of learning from data, that is

$$\mathcal{P}(\boldsymbol{f}|\mathcal{D}) = \frac{\mathcal{P}(\mathcal{D}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})}{\int \mathcal{P}(\mathcal{D}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})\, d\boldsymbol{f}} = \frac{\mathcal{P}(\mathcal{D}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})}{E[\mathcal{P}(\mathcal{D}|\boldsymbol{f})]_0} \tag{3.55}$$

where $E[\mathcal{P}(\mathcal{D}|\boldsymbol{f})]_0$ is the average of the likelihood with respect to the prior distribution $\mathcal{P}(\boldsymbol{f})$. The posterior distribution could be used to not only express posterior expectations as typically high dimensional integrals, but also account for the joint distribution along with the test samples in making prediction.

Suppose that the target values in the training samples $\mathcal{D}$ have been contaminated by a zero-mean Gaussian noise with variance σ^2, i.e. $y_i = f(x_i) + \delta_i \ \forall i$. Since the noise δ_i is independent Gaussian and the function values $\boldsymbol{f}$ have a joint Gaussian as given in (3.54), the target values $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^T$ have a joint Gaussian with mean $\boldsymbol{f}_0$ and variance matrix $\Sigma + \sigma^2\mathbf{I}$ where $\mathbf{I}$ is a $n \times n$ identity matrix. Now let us consider the joint probability together with m test samples $\boldsymbol{f}_t$ that is indexed by x without target values, which can be written as

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_t \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f}_0 \\ \boldsymbol{f}_{t0} \end{bmatrix}, \begin{bmatrix} \Sigma + \sigma^2\mathbf{I} & \boldsymbol{k} \\ \boldsymbol{k}^T & \Sigma_t \end{bmatrix}\right) \tag{3.56}$$

where $\boldsymbol{f}_{t0}$ denotes the prior mean for test samples, Σ_t denotes the $m \times m$ covariance matrix of

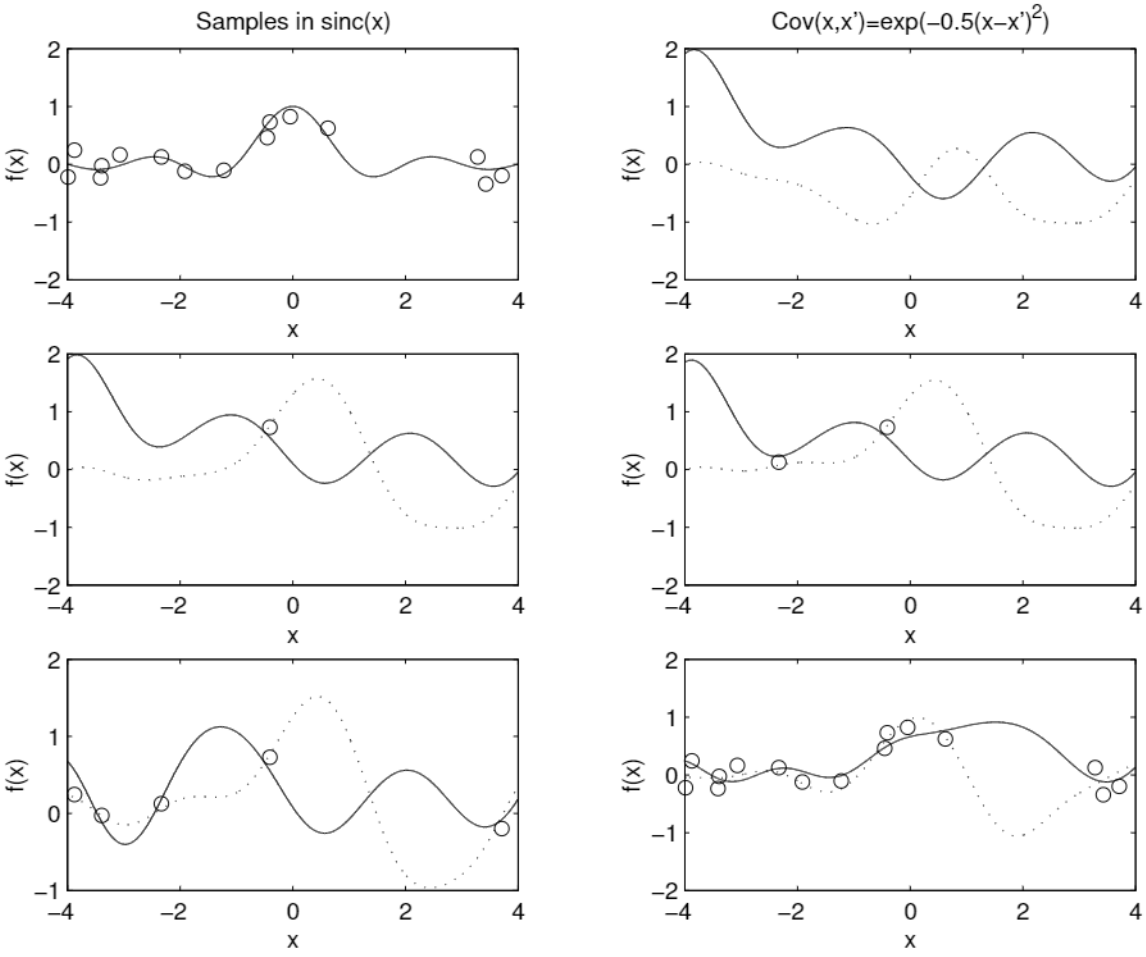

Figure 3.8: Samples drawn from the posterior distribution of the zero-mean Gaussian process defined with covariance function $Cov(x, x') = \exp\left(-\frac{1}{2}(x - x')^2\right)$. The circles denotes the training pairs we drew from $\text{sinc}(x)$ in presence of additive zero-mean Gaussian noise with variance 0.2^2. Five cases $(b) \sim (f)$ are presented along with the underlying function (a), in which we are given the 0, 1, 2, 5 and 15 training samples accordingly.

test samples and $\boldsymbol{k}$ denotes their $n \times m$ correlation matrix. What we actually concern is the conditional distribution of $\boldsymbol{f}_t$ given $\boldsymbol{y}$, $\mathcal{P}(\boldsymbol{f}_t|\boldsymbol{y})$ which is also a Gaussian as

$$P(\boldsymbol{f}_t|\boldsymbol{y}) \sim \mathcal{N}\left(\boldsymbol{f}_{t0} + \boldsymbol{k}^T(\Sigma + \sigma^2\mathbf{I})^{-1}\boldsymbol{y}, \Sigma_t - \boldsymbol{k}^T(\Sigma + \sigma^2\mathbf{I})^{-1}\boldsymbol{k}\right) \tag{3.57}$$

We use a simple example to illustrate the learning process in Gaussian processes. Suppose we are given 15 training samples by sampling from the function $\text{sinc}(x)$. The additive noise is a zero-mean Gaussian random variable with variance 0.2^2. We choose a zero-mean Gaussian process with covariance function $Cov(x, x') = \exp\left(-\frac{1}{2}(x - x')^2\right)$ to account for the training data. We study a serial cases of sampling the posterior distribution of the Gaussian process and

present their results in Figure 3.8. The training samples and the underlying generator are given in Figure 3.8(a). Before any training data arrive, we sample the Gaussian process twice and present the curve in Figure 3.8(b), where the input x (i.e. the test samples) was 801 linearly equally spaced points between -4 and $+4$. This graph is same as the top-left graph in Figure 3.6. Now suppose we are given one training sample, the corresponding sampling result is shown in Figure 3.8(c). The case that we are given another sample is presented in Figure 3.8(d). Notice the change of the two curves nearby the training samples. The cases of sampling in the posterior distribution of the Gaussian process given 5 and 15 training samples are shown in Figure 3.8(e) and (f) separately. Notice that the sampling results in the posterior distribution tend to be better approximations of the underlying function, as we are given more training samples.

3.2.3 Predictive Distribution

In making prediction, we might be interested in expectation and variance of function values at some particular input points x in the Gaussian process. The following lemma (Csató and Opper, 2002) shows that simple but important predictive quantities like the posterior mean and the posterior variance of the process at arbitrary inputs can be expressed as a linear combination of a finite set of parameters which depend on the training data only. For arbitrary likelihoods, the following can be shown.

Parameterization Lemma (Csató and Opper, 2002). The result of the Bayesian learning (3.55) using a Gaussian process prior with mean $f_0(x)$ and the covariance $Cov(x, x')$ and data $\mathcal{D} = \{(x_i, y_i)|i = 1, \ldots, n\}$ is a process with mean and kernel functions given by

$$E[f(x)]_n = f_0(x) + \sum_{i=1}^{n} Cov(x, x_i) \cdot w(i) \tag{3.58}$$

$$Cov(x, x')_n = Cov(x, x') + \sum_{i,j=1}^{n} Cov(x, x_i) \cdot R(ij) \cdot Cov(x_j, x') \tag{3.59}$$

where the parameters $w(i)$ and $R(ij)$ are given by

$$w(i) = \frac{\partial}{\partial f_0(x_i)} \ln \int \mathcal{P}(\mathcal{D}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})d\boldsymbol{f} \tag{3.60}$$

$$R(ij) = \frac{\partial^2}{\partial f_0(x_i)\partial f_0(x_j)} \ln \int \mathcal{P}(\mathcal{D}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})d\boldsymbol{f} \tag{3.61}$$

□

The parameters $w(i)$ and $R(ij)$ have to be computed during the training of the GP model, but once only, and then are fixed when we make predictions. Notice that the parametric form of the posterior mean is consistent with the representations for the predictors in other kernel methods, such as support vector machines (1.23), if we assume a common prior mean for all the function values that resembles the bias b in (1.23). While the latter representations are derived from the celebrated representer theorem (Kimeldorf and Wahba, 1971; Schölkopf et al., 2001), the Parameterization Lemma could be derived from properties of Gaussian processes directly. As for a proof, see the Appendix D or Csató and Opper (2002).

As the conjugate case, the posterior distribution of Gaussian process is also Gaussian if Gaussian noise model is used in likelihood, and then can be derived analytically as in (1.26) where zero mean is assumed. For a likelihood other than Gaussian, the posterior process is in general not Gaussian and the integrals cannot be computed in a closed form. Hence, we have to resort to approximations to keep the inference tractable (Csató et al., 2000). A popular method is to approximate the posterior by a Gaussian distribution (Williams and Barber, 1998). This may be formulated within a variational approach, where a certain dissimilarity measure between the true and the approximate distribution is minimized (Seeger, 1999).

3.2.4 On-line Formulation

Another interesting fact that needs to be pointed out here is the on-line formulation (Csató and Opper, 2002), which is a sequential mode to learn from the training data by sweeping through the samples once only. In order to compute the on-line approximations of the mean and covariance, we apply Parameterization Lemma sequentially with only one likelihood term $P(y_t|x_t)$ at an iteration step. Proceeding recursively, we arrive at

$$E[f(x)]_{k+1} = E[f(x)]_k + w(k+1)Cov(x, x_{k+1})_k \tag{3.62}$$

$$Cov(x, x')_{k+1} = Cov(x, x')_k + r(k+1)Cov(x, x_{k+1})_k Cov(x_{k+1}, x')_k \tag{3.63}$$

where $E[\cdot]_k$ is the average with respect to the Gaussian process given k training samples, i.e.

$$\begin{aligned} E[f(x)]_k &= \int f(x)\mathcal{P}(f(x)|y_1, \ldots, y_k)\, df(x), \\ Cov(x, x_{k+1})_k &= E\left[(f(x) - E[f(x)]_k)(f(x_{k+1}) - E[f(x_{k+1})]_k)\right]_k, \end{aligned} \tag{3.64}$$

and the parameters $w(k+1)$ and $r(k+1)$ are given by

$$\begin{aligned} w(k+1) &= \frac{\partial}{\partial E\big[f(x_{k+1})\big]_k} \ln E[\mathcal{P}\big(y_{k+1}|f(x_{k+1})\big)\big]_k \\ r(k+1) &= \frac{\partial^2}{\partial E[f(x_{k+1})]_k^2} \ln E\big[\mathcal{P}\big(y_{k+1}|f(x_{k+1})\big)\big]_k \end{aligned} \tag{3.65}$$

We give a simple example to illustrate the on-line version of the Parameterization Lemma in Gaussian processes. Suppose we have drawn 15 training samples by sampling from the function $\mathrm{sinc}(x)$, which are same as we had done in the previous section. The additive noise is zero-mean Gaussian with variance $\sigma^2 = 0.2^2$. We choose a zero-mean Gaussian process with covariance function $Cov(x, x') = \exp\left(-\frac{1}{2}(x - x')^2\right)$ to account for the training data. We study a serial posterior distribution of the Gaussian process and present the posterior mean and variance in Figure 3.9. The training samples and the underlying generator are given in Figure 3.8(a). Now suppose we are given the first training sample, the corresponding posterior distribution is shown in Figure 3.9(a). The case that we have been given the second sample is presented in Figure 3.9(b). Notice the change of the posterior mean and the error bar nearby the training samples. The posterior distribution of the Gaussian process given 5 and 15 training samples are shown in Figure 3.9(c) and (d) accordingly. Since the Gaussian noise model is used, this is a conjugate case. For batch mode, the specific formulation has been given in (1.26) that can be derived analytically from the Parameterization Lemma. As for sequential mode, the parameters in on-line update formulation (3.61) can be specified as

$$\begin{aligned} w(k+1) &= \frac{y_{k+1} - E[f(x_{k+1})]_k}{\varsigma_k^2} \\ r(k+1) &= -\frac{1}{\varsigma_k^2} \end{aligned} \tag{3.66}$$

where $\varsigma_k^2 = \sigma^2 + Cov(x_{k+1}, x_{k+1})_k$. We start at $k = 0$ with $E[f(x_1)]_0 = 0$ and $Cov(x_1, x_1)_0 = 1$ to calculate $w(1)$ and $r(1)$ by (3.66). We apply the formulation (3.62) and (3.63) to calculate current mean and covariance whenever it is needed. Then $k = k + 1$, repeat this procedure till $k = n$. Notice that there is an equivalence in learning result between the batch mode and the sequential mode. However, the sequential mode does not require any matrix inverse that makes it possible to be interleaved with sparsification steps (Csató and Opper, 2002) for large data sets.

3.2.5 Determining the Hyperparameters

We have set up a model for the training data $\mathcal{D}$ we are given. We now follow a consistent way to deal with the undetermined hyperparameter vector Θ in the model, which is composed of the parameters in covariance function and the parameters in likelihood function. Ideally, we would like to integrate over all the hyperparameters in order to make predictions, i.e.

$$\mathcal{P}(f(x)|x,\mathcal{D}) = \int \mathcal{P}(f(x)|x,\mathcal{D},\Theta)\mathcal{P}(\Theta|\mathcal{D})d\Theta \tag{3.67}$$

Here, we can get approximations to the posterior distribution (3.67) by either approximating the average prediction using the most probable values of the hyperparameters Θ (MacKay, 1992c; Williams and Rasmussen, 1996), which approach is referred to as evidence maximization, or by performing the integration over the Θ space numerically using Monte Carlo methods (Neal, 1997a).

3.2.5.1 Evidence Maximization

Evidence maximization uses an approximation to the integral in (3.67) based on the most probable set of hyperparameters Θ_{MP}:

$$\mathcal{P}(f(x)|x,\mathcal{D}) \simeq \mathcal{P}(f(x)|x,\mathcal{D},\Theta_{\mathrm{MP}}) \tag{3.68}$$

where $\Theta_{\mathrm{MP}} = \arg\max_{\Theta} \mathcal{P}(\Theta|\mathcal{D})$. The posterior distribution can be written as

$$\mathcal{P}(\Theta|\mathcal{D}) = \frac{\mathcal{P}(\mathcal{D}|\Theta)\mathcal{P}(\Theta)}{\mathcal{P}(\mathcal{D})} \tag{3.69}$$

The dominator is independent of Θ and could be ignored in finding Θ_{MP}. The two remaining terms, the likelihood of Θ and the prior on Θ, shall be considered in terms of their logs for computational convenience. For the conjugate case with Gaussian likelihood, the log likelihood and its derivatives have been given as in (1.28) and (1.29) respectively. As we typically have little knowledge about the suitable values of Θ before training data are available, we usually assume a vague distribution for $\mathcal{P}(\Theta)$ that is greatly insensitive to the value Θ. Therefore, it is common practice to ignore the log prior term and perform a maximum likelihood optimization of the hyperparameters Θ (MacKay, 1992c). The evidence $\mathcal{P}(\mathcal{D}|\Theta)$ can be used to assign a preference to alternative values of the hyperparameters Θ. When the derivatives of the evidence $\mathcal{P}(\mathcal{D}|\Theta)$ with respect to θ can be derived, we can search for Θ_{MP} by some standard gradient-based

optimization packages.

3.2.5.2 Monte Carlo Approach

The Markov chain Monte Carlo approach (MCMC) (Neal, 1993) uses sampling methods to calculate an approximation to the predictive distribution. The MCMC approach constructs a Markov chain to approximate the posterior distribution $P(\Theta|\mathcal{D})$ in which each sample depends on the previous one as well as having a random component, and then the predictive distribution (3.67) could be approximated by a mixture of the samples in the Markov chain, we can write

$$\mathcal{P}(f(x)|x,\mathcal{D}) \simeq \frac{1}{M}\sum_{t=1}^{M}\mathcal{P}(f(x)|x,\mathcal{D},\Theta_t) \tag{3.70}$$

where the Θ_t are samples in the Markov chain that approximates the posterior distribution over Θ, $\mathcal{P}(\Theta|\mathcal{D})$. Note that as we are sampling from the posterior distribution $\mathcal{P}(\Theta|\mathcal{D})$, we shall need priors on these hyperparameters $\mathcal{P}(\Theta)$.

As for the specification of the prior $\mathcal{P}(\Theta)$, we choose the popular family of covariance function (3.53) as an example. Following the suggestions in Rasmussen (1996), we usually collect $\{\ln\kappa_0, \ln\kappa_a, \ln\kappa_b\}$ and $\{\ln\kappa_\iota\}_{\iota=1}^{d}$ as variables to tune.[13] The prior assumes that the training data has been normalized to roughly zero mean and unit variance. The priors on $\ln\kappa_a$ and $\ln\kappa_b$ are all Gaussian with mean -3 and standard deviation 3, i.e. $\mathcal{N}(-3,3)$. Since the targets are assumed to be normalized to roughly unit variance, we expect the hyperparameter $\ln\kappa_0$ to be in the vicinity of 0. So a reasonable prior on $\ln\kappa_0$ is a Gaussian with mean -1 and standard deviation 1, i.e. $\mathcal{N}(-1,1)$. The priors for ARD hyperparameters are slightly complicated. Following the derivations in Neal (1996) and Rasmussen (1996), we could use a Gamma prior for the ARD hyperparameters $\{\ln\kappa_\iota\}_{\iota=1}^{d}$, which is

$$\mathcal{P}(\ln\kappa_\iota) = d^{-1}\frac{(\alpha/2\mu_0)^{\alpha/2}}{\Gamma(\alpha/2)}\exp\left(\frac{\alpha\ln\kappa_\iota}{2} - \frac{\alpha\exp(\ln\kappa_\iota)}{2\mu_0 d^{2/\alpha}}\right) \tag{3.71}$$

The parameter μ_0 is usually fixed at 1.[14] To make the prior non-informative (i.e. vague), we might fix α to small values. The smaller α is, the vaguer the prior becomes. As an extreme limit, by setting α to zero, a uniform hyperprior is obtain, i.e. $\mathcal{P}(\ln\kappa_\iota) \propto 1$.

To construct the Markov chain effectively, we can take account of information concerning

[13] The hyperparameters are constrained to be positive. Logs of these hyperparameters convert the optimization for finding Θ_{MP} into an unconstrained optimization problem that is more convenient for standard optimization packages.

[14] We can also try to make μ_0 as a top level hyperparameter and set some vague prior on it as did in (Neal, 1997a).

the gradient of $\mathcal{P}(\Theta|\mathcal{D})$ and use this to choose search directions which favour regions of high probability. A procedure to achieve this, known as hybrid Monte Carlo, was developed by Duane et al. (1987), and has been successfully applied by Rasmussen (1996) and Neal (1997a) to implement Gaussian processes.

3.2.5.3 Evidence vs Monte Carlo

In the approach of evidence maximization, a potential difficulty lies in the posterior distribution $\mathcal{P}(\Theta|\mathcal{D})$ might be multi-modal. This could imply that the Θ_{MP} found by the gradient-based optimization package is dependent on the initial Θ used. In general we can train several times starting from different initial states, and choosing the one with the highest probability 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. Another issue we need notice is the computational cost. For standard Gaussian processes for regression in which Gaussian likelihood is used, each evaluation of the gradient of the log likelihood requires the evaluation of $\mathbf{H}^{-1}$ as in (1.29). Any exact inversion method has an associated computational cost that is $\mathcal{O}(n^3)$ and so calculating gradients exactly becomes time consuming for large training data sets (more than 1000 samples).

Using MCMC, we approximate our posterior probabilities $\mathcal{P}(\Theta|\mathcal{D})$ using average over a series of samples. For collecting each sample in the series, we have to compute all the information required by the Monte Carlo simulation, in which inverting the covariance matrix is needed every time but without the need to store this inverse.[15] When we make predictions, unless we have retain all the inverses, we must go back to re-compute them at considerable expense. Thus, it is very expensive for the Monte Carlo approach to tackle large data sets. However, the Monte Carlo approach does offer us significantly more flexibility. Non-Gaussian noise model can be incorporated into Gaussian processes, that is essential for classification problems.

For smaller data sets where matrix storage and inverting are not an important issue, we believe that the Monte Carlo approach could give better results than evidence maximization in a reasonable amount of CPU time. For large data sets where the multi-modality in $\mathcal{P}(\Theta|\mathcal{D})$ becomes not acute, the approach of evidence maximization is fast and their performance is found competitive (Rasmussen, 1996).

[15] It is very expensive to store a series of $n \times n$ matrices in memory.

3.3 Some Relationships

We have introduced a Bayesian framework on neural networks from the weight-space view, and another framework in Gaussian processes from the function-space view. Although these two Bayesian frameworks are quite different in formulation, there are some relationships and equivalences between them.

3.3.1 From Neural Networks to Gaussian Processes

As shown by Neal (1996), there is a strong relationship between Gaussian processes and neural networks. Let us consider the MLP networks with one hidden layer (see Figure 3.2), which could be mathematically described as:

$$f(x) = \sum_{i=1}^{m} w_i z_i(x) + b \tag{3.72}$$

with $z_i(x) = \varphi\left(\sum_{j=1}^{d} w_i^j \cdot x^j\right)$ where the activation function $\varphi(\cdot)$ is usually the tanh or logistic function. Following MacKay (1992c), we shall specify Gaussian priors on all the weights. Each Gaussian has zero mean and standard deviation σ_d, σ_h and σ_b for the input-to-hidden weights w_i^j, the hidden-to-output weights w_i and the bias w_0 respectively. For an arbitrary input, x, the expectation of the output $E(f(x)) = \sum_{i=0}^{m} E(w_i)E(z_i)$ is zero since $E(w_i) = 0$ and the independence between w_i and w_i^j by our assumption. The variance of $f(x)$ can be written as

$$E\big(f^2(x)\big) = E\left(\left(\sum_{i=1}^{m} w_i z_i + b\right)^2\right) = m\sigma_h^2 E(z_i^2) + \sigma_b^2 \tag{3.73}$$

as the weights are independent and $E(w_i^2) = \sigma_h^2$. We can use the Central Limit Theorem (Bickel and Doksum, 1977) and the fact that $E(z_i^2)$ is the same for all i to conclude that for large m the total contribution of the hidden units to the output value $f(x)$ becomes approximately Gaussian with variance $m\sigma_h^2\mathcal{C}(x,x)$ where we define $\mathcal{C}(x,x) = E(z_i^2)$. The bias, w_0, is also a Gaussian, with variance σ_b^2, so for large m the prior distribution of $f(x)$ is also approximately a Gaussian, with variance $\sigma_b^2 + m\sigma_h^2\mathcal{C}(x,x)$. We would like the variance to be finite even for an infinite number of hidden neurons. $\mathcal{C}(x,x)$ is finite since the z_i is bounded. If we scale the prior variance of the hidden-to-output weights w_i according to the number of hidden neurons, setting $\sigma_h = \varpi_h m^{-1/2}$ in which ϖ_h is a constant, then the variance becomes $\varpi_h^2\mathcal{C}(x,x) + \sigma_b^2$. We can use the similar argument to investigate the prior joint distribution of the values of output for the inputs, i.e. the joint distribution of $f(x_1), \ldots, f(x_n)$ where $x_1, \ldots, x_n$ are the input vectors. As m goes

to infinity, the prior joint distribution converges to a multivariate Gaussian, with zero mean and covariance $\varpi_h^2 \mathcal{C}(x_k, x_l) + \sigma_b^2$ where $\mathcal{C}(x_k, x_l) = E\big(z_i(x_k) z_i(x_l)\big)\ \forall k, l$ which is some function dependent on σ_d and the outputs of the hidden neuron z_i. A parallelling argument could be straightforwardly extended to the model of fixed basis functions (Gibbs, 1997). Distributions over functions of this sort, in which the joint prior distribution of the values of the function at any finite number of points is multivariate Gaussian, are known as *Gaussian processes*.

3.3.2 Between Weight-space and Function-space

Suppose the regression function contributed by some fixed basis function networks possesses the form $f(x) = \sum_{i=1}^{m} w_i \phi_i(x) = \mathrm{w}^T \boldsymbol{\phi}(x)$, where w is the weight vector and $\boldsymbol{\phi}(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_m(x)]^T$ is composed of a set of m basis functions $\{\phi_i(x)\}_{i=1}^{m}$ (m might be infinite). Let the weights have a prior distribution which is Gaussian and centered on the origin, $\mathrm{w} \sim \mathcal{N}(0, \Sigma_{\mathbf{w}})$, in which the covariance matrix $\Sigma_{\mathbf{w}}$ is a diagonal matrix, usually an identity matrix. Again, assuming that the targets y_i are corrupted by Gaussian noise with variance σ^2, the likelihood of w is $\mathcal{P}(y_1, \ldots, y_n | \mathrm{w}) = \frac{1}{(2\pi\sigma^2)^{n/2}} \prod_{i=1}^{n} \exp\left(-\frac{(y_i - f(x_i;\mathrm{w}))^2}{2\sigma^2}\right)$. The posterior mean value of the weights w_{MP} is found by

$$\arg\min_{\mathrm{w}} \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - f(x_i;\mathrm{w}))^2 + \frac{1}{2} \mathrm{w}^T \Sigma_{\mathbf{w}}^{-1} \mathrm{w} \tag{3.74}$$

It is easy to see that $\mathrm{w}_{\mathrm{MP}} = (\frac{1}{\sigma^2}\Sigma_{\mathbf{w}}^{-1} + \Phi^T\Phi)^{-1}\Phi^T \boldsymbol{y}$, where Φ is the $n \times m$ matrix with ij-th entry $\phi_j(x_i)$ and $\boldsymbol{y} = [y_1, \ldots, y_n]^T$. The regression function with w_{MP} shall be $f(x; \mathrm{w}_{\mathrm{MP}}) = \boldsymbol{\phi}^T(x)(\frac{1}{\sigma^2}\Sigma_{\mathbf{w}}^{-1} + \Phi^T\Phi)^{-1}\Phi^T \boldsymbol{y}$, which is also the predictive mean at the input x given by the fixed basis function network. Note that $(\frac{1}{\sigma^2}\Sigma_{\mathbf{w}}^{-1} + \Phi^T\Phi)^{-1}\Phi^T = \Sigma_{\mathbf{w}}\Phi^T(\Phi\Sigma_{\mathbf{w}}\Phi + \sigma^2\mathbf{I})^{-1}$ holds. Therefore the mean of the predictive distribution can be written as

$$f(x; \mathrm{w}_{\mathrm{MP}}) = \boldsymbol{\phi}^T(x)\Sigma_{\mathbf{w}}\Phi^T(\Phi\Sigma_{\mathbf{w}}\Phi^T + \sigma^2\mathbf{I})^{-1}\boldsymbol{y}$$

The predictive variance $Var[f(x;\mathrm{w})]$, which is also called the "error bar" of the prediction at x, is given by

$$\begin{aligned} Var[f(x;\mathrm{w})] &= E_{\mathbf{w}|\mathcal{D}}[(f(x;\mathrm{w}) - f(x;\mathrm{w}_{\mathrm{MP}}))^2] \\ &= \boldsymbol{\phi}^T(x) E_{\mathbf{w}|\mathcal{D}}[(\mathrm{w} - \mathrm{w}_{\mathrm{MP}})(\mathrm{w} - \mathrm{w}_{\mathrm{MP}})^T]\boldsymbol{\phi}(x) \\ &= \boldsymbol{\phi}^T(x)(\tfrac{1}{\sigma^2}\Sigma_{\mathbf{w}}^{-1} + \Phi^T\Phi)^{-1}\boldsymbol{\phi}(x) \\ &= \boldsymbol{\phi}^T(x)\Sigma_{\mathbf{w}}\boldsymbol{\phi}(x) - \boldsymbol{\phi}^T(x)\Sigma_{\mathbf{w}}\Phi^T(\Phi\Sigma_{\mathbf{w}}\Phi^T + \sigma^2\mathbf{I})\Phi\Sigma_{\mathbf{w}}\boldsymbol{\phi}(x) \end{aligned}$$

Now let us look at the predictive mean of standard Gaussian processes given in (1.26), that is $\mathbf{k}^T(\Sigma + \sigma^2\mathbf{I})^{-1}\boldsymbol{y}$, where $\mathbf{k} = [Cov(x_1, x), Cov(x_2, x), \ldots, Cov(x_n, x)]^T$ and Σ is the $n \times n$ covariance matrix whose ij-th element is $Cov(x_i, x_j)$, and the predictive variance is $Cov(x, x) + \mathbf{k}^T(\Sigma + \sigma^2\mathbf{I})^{-1}\mathbf{k}$. We notice that there is an equivalence in these two regression framework provided that $Cov(x_i, x_j) = \boldsymbol{\phi}^T(x_i)\Sigma_{\mathbf{w}}\boldsymbol{\phi}(x_j)$. In other words, the Bayesian neural network with fixed basis function $\{\phi_i(x)\}$ in the weight-space yields the same regression formulation as the standard Gaussian process defined by the covariance function $\boldsymbol{\phi}^T(x_i)\Sigma_{\mathbf{w}}\boldsymbol{\phi}(x_j)$ does.

Given a positive definite covariance function $Cov(x_i, x_j)$ (or kernel function), it is possible to find out such a basis function in reproducing kernel Hilbert space. According to Mercer-Hilbert-Schmidt theorem (Wahba, 1990; Riesz and Sz.-Nagy, 1955), we can get an orthonormal sequence of continuous eigenfunctions, $\phi_1, \phi_2, \ldots$ and eigenvalues $\upsilon_1 \geq \upsilon_2 \geq \ldots \geq 0$, with $Cov(x_i, x_j) = \sum_{\tau=1}^{\infty} \upsilon_\tau \phi_\tau(x_i)\phi_\tau(x_j)$. Simply choosing $\upsilon_\tau^{1/2}\phi_\tau(x)$ as the basis function and an identity matrix as $\Sigma_{\mathbf{w}}$, we can find the equivalent basis function network in weight-space for any given Gaussian processes, which might be of infinite hidden neurons. It is also the direct approach to probabilistic framework for support vector machines.

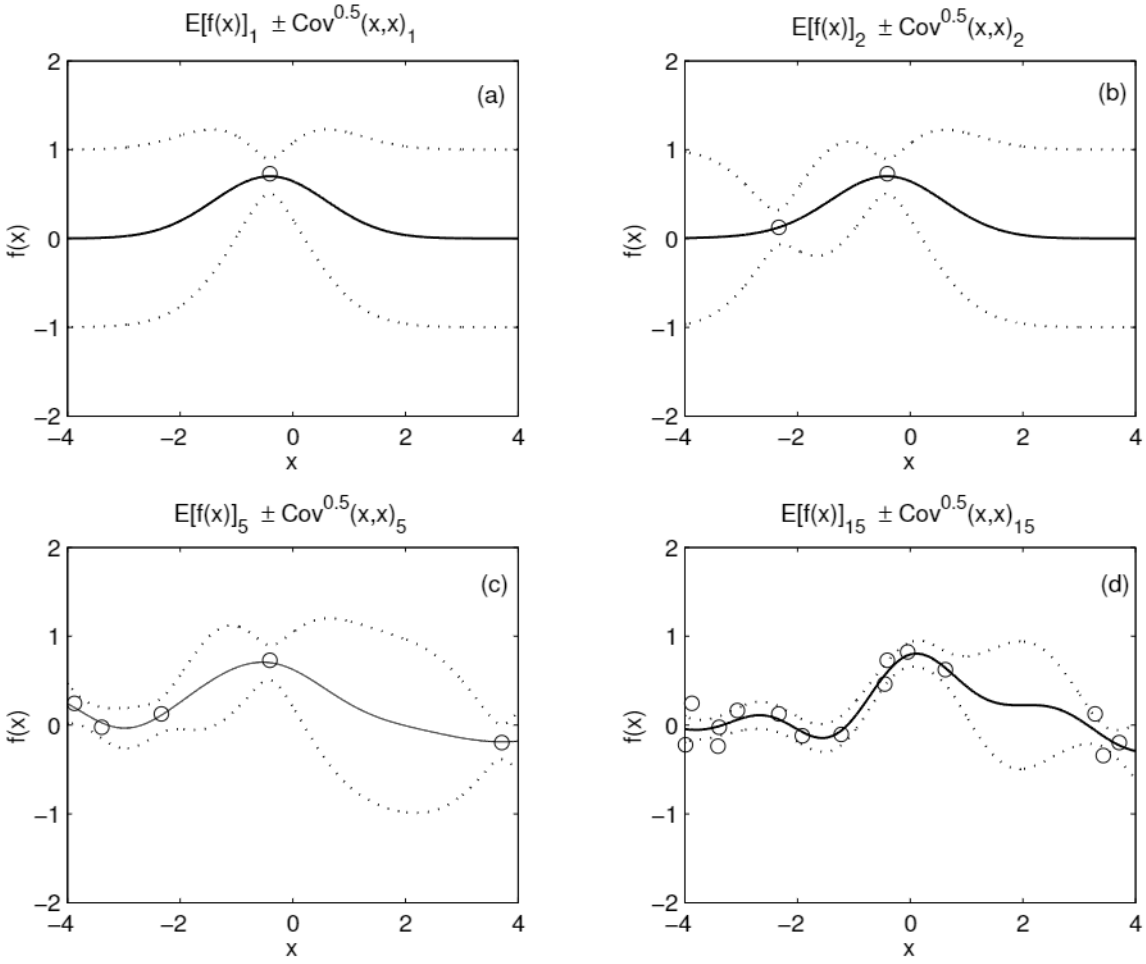

Figure 3.9: The mean and its variance of the posterior distribution of the Gaussian process given k training samples. Four cases $(a) \sim (d)$ are presented in which we are given 1, 2, 5 and 15 training samples accordingly. The prior distribution of the Gaussian process is defined with zero mean and covariance function $Cov(x, x') = \exp\left(-\frac{1}{2}(x - x')^2\right)$. The circles denote the training pairs we drew from $\text{sinc}(x)$ in presence of additive zero-mean Gaussian noise with variance 0.2^2. The solid curves are the posterior means $E[f(x)]_k$, and the dotted curves are the error bars, i.e. $E[f(x)]_k \pm \sqrt{Cov(x, x)_k}$.

Chapter 4

Bayesian Support Vector Regression

The application of Bayesian techniques to neural networks was pioneered by Buntine and Weigend (1991), MacKay (1992c) 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 advantage of GPR models over other non-Bayesian models is the explicit probabilistic formulation. This not only builds the ability to infer hyperparameters in Bayesian framework but also provides confidence intervals in prediction. The drawback of GPR models lies in the huge computational cost for large data sets.

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: a global minimum solution as the minimization of a convex programming problem; relatively fast training speed; and sparseness in solution representation. The performance of SVR cru-

cially 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 parameters 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, 1992c) using a weight-space interpretation. Seeger (1999) 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 (2001) 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 employ a unified non-quadratic loss function for SVR, called the soft insensitive loss function (SILF). The SILF is C^1 smooth. Further, it retains 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 bar can also be provided in making predictions.

This chapter is organized as follows: in section 4.1 we review the standard framework of regression with Gaussian processes, and employ the SILF as loss function in likelihood evaluation; in section 4.2 we formulate the MAP estimate on the function values as a convex quadratic programming problem; hyperparameter inference is discussed in section 4.3 and predictive distribution is discussed in section 4.4; in section 4.5 we show the results of numerical experiments that verify the approach.

4.1 Probabilistic Framework

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

$$y_i = f(x_i) + \delta_i \qquad i = 1, 2, \ldots, n \tag{4.1}$$

where the δ_i are independent, identically distributed (i.i.d.) random variables, whose distributions are usually unknown. Regression aims to infer the function $\boldsymbol{f}$, or an estimate of it, from the finite data set $\mathcal{D}$. In the Bayesian approach, we regard the function $\boldsymbol{f}$ as the realization of a random field with a known prior probability. The posterior probability of $\boldsymbol{f}$ given the training data $\mathcal{D}$ can then be derived by Bayes' theorem:

$$\mathcal{P}(\boldsymbol{f}|\mathcal{D}) = \frac{\mathcal{P}(\mathcal{D}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})}{\mathcal{P}(\mathcal{D})} \tag{4.2}$$

where $\boldsymbol{f} = [f(x_1), f(x_2), \ldots, f(x_n)]^T$, $\mathcal{P}(\boldsymbol{f})$ 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, 1998; Williams and Barber, 1998) to describe a Bayesian framework.

4.1.1 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 possible choices of covariance function have been discussed in Section 3.2.1. We prefer the Gaussian covariance function which is defined as

$$Cov[f(x_i), f(x_j)] = Cov(x_i, x_j) = \kappa_0 \exp\left(-\frac{\kappa}{2}\sum_{l=1}^{d}(x_i^l - x_j^l)^2\right) + \kappa_b \tag{4.3}$$

where $\kappa > 0$, $\kappa_0 > 0$ denotes the average power of $f(x)$, $\kappa_b > 0$ denotes the variance of the offset to the function $f(x)$, and x^ι denotes the ι-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 (4.3) is the Gaussian kernel in SVM, while the second term corresponds to the variance of the bias in classical SVR (Vapnik, 1995). Compared with the covariance function (3.53) used in standard Gaussian processes designs (Williams, 1998; Williams and Barber, 1998), the Gaussian covariance function (4.3) does not have the linear component. We drop off the linear term due to the fact that there is some correlation between the parameters κ_0 and κ_a in (3.53) which might produce multiple solutions in hyperparameter inference. In the cases that we believe there is some linear trend in the data, we can use linear covariance function (3.51)

only to remove the linear trend from the data as a preprocessing.

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

$$\mathcal{P}(\boldsymbol{f}) = \frac{1}{\mathcal{Z}_{\boldsymbol{f}}} \exp\left(-\frac{1}{2}\boldsymbol{f}^T \Sigma^{-1} \boldsymbol{f}\right) \tag{4.4}$$

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]

4.1.2 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 (4.1) is i.i.d. with probability distribution $\mathcal{P}(\delta_i)$, $\mathcal{P}(\mathcal{D}|\boldsymbol{f})$ can be evaluated by:

$$\mathcal{P}(\mathcal{D}|\boldsymbol{f}) = \prod_{i=1}^{n} \mathcal{P}(y_i - f(x_i)) = \prod_{i=1}^{n} \mathcal{P}(\delta_i) \tag{4.5}$$

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

$$\mathcal{P}(\delta_i) \propto \exp(-C \cdot \ell(\delta_i)) \tag{4.6}$$

where $\ell(\cdot)$ is called the loss function and C is a parameter greater than zero.

In standard GPR (Williams and Rasmussen, 1996; Williams, 1998), Gaussian noise model (2.8) is used as the likelihood function $\ell(\cdot)$. The Gaussian process prior for the functions $\boldsymbol{f}$ is conjugated with the Gaussian likelihood to yield a posterior distribution over functions that can be used in hyperparameter inference and prediction. The posterior probability over functions can be carried out exactly using matrix operations in the GPR formulation. This is one of the reasons that the Gaussian noise model is popularly used. However, one of the potential disadvantages 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 (see Section 2.1.2 for more details). Soft insensitive loss function (SILF) has been introduced in Section 2.2.1 as a unified non-quadratic loss function. SILF possesses several advantages, such as insensitivity to outliers and sparseness

[1] If the covariance is defined using (4.3), Σ is symmetric and positive definite if $\{x_i\}$ is a set of distinct points in $\mathbb{R}^d$ (Micchelli, 1986).

in solution representation. Moreover, it is C^1 smooth. The definition of SILF (2.28) is given as

$$\ell_{\epsilon,\beta}(\delta) = \begin{cases} |\delta| - \epsilon & if \quad |\delta| > (1+\beta)\epsilon \\ \dfrac{(|\delta| - (1-\beta)\epsilon)^2}{4\beta\epsilon} & if \quad (1+\beta)\epsilon \geq |\delta| \geq (1-\beta)\epsilon \\ 0 & if \quad |\delta| < (1-\beta)\epsilon \end{cases} \tag{4.7}$$

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

Using SILF, the likelihood function is written as

$$\mathcal{P}(\mathcal{D}|\boldsymbol{f}) = \frac{1}{\mathcal{Z}_S^n} \exp\left(-C \sum_{i=1}^{n} \ell_{\epsilon,\beta}\left(y_i - f(x_i)\right)\right) \tag{4.8}$$

where $\ell_{\epsilon,\beta}(\cdot)$ and $\mathcal{Z}_S$ are defined as in (2.31) and (2.32) respectively. The loss function characterizes the noise distribution, which together with the prior probability $\mathcal{P}(\boldsymbol{f})$, determines the posterior probability $\mathcal{P}(\boldsymbol{f}|\mathcal{D})$ via Bayes' theorem.

4.1.3 Posterior Probability

Based on Bayes' theorem (4.2), prior probability (4.4) and the likelihood (4.8), the posterior probability of $\boldsymbol{f}$ can be written as

$$\mathcal{P}(\boldsymbol{f}|\mathcal{D}) = \frac{1}{\mathcal{Z}} \exp\left(-S(\boldsymbol{f})\right) \tag{4.9}$$

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

$$\min_{\boldsymbol{f}} S(\boldsymbol{f}) = C \sum_{i=1}^{n} \ell_{\epsilon,\beta}\left(y_i - f(x_i)\right) + \frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f} \tag{4.10}$$

Let $\boldsymbol{f}_{\text{MP}}$ be the optimal solution of (4.10). Since the SILF is differentiable, the derivative of $S(\boldsymbol{f})$ with respect to $\boldsymbol{f}$ should be zero at $\boldsymbol{f}_{\text{MP}}$, i.e.

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

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

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

$$\boldsymbol{f}_{\mathrm{MP}} = \Sigma \cdot \boldsymbol{w} \tag{4.11}$$

We can further decompose the solution (4.11) into the form

$$\boldsymbol{f}_{\mathrm{MP}}(x) = \sum_{i=1}^{n} w_i \cdot \kappa_0 \cdot K(x, x_i) + \kappa_b \sum_{i=1}^{n} w_i = \kappa_0 \sum_{i=1}^{n} w_i \cdot K(x, x_i) + b \tag{4.12}$$

where $b = \kappa_b \sum_{i=1}^{n} w_i$ and $K(x, x_i) = \exp\left(-\frac{\kappa}{2} \sum_{l=1}^{d} (x^l - x_i^l)^2\right)$ is just the Gaussian kernel in classical SVR, to show the significance of the hyperparameter in regression function. The contribution of each pattern to the regression function depends on its w_i in (4.12), and κ_0 is the average power of all the training patterns. κ_b is only involved in the bias term b in (4.12) and that might be trivial if the sum $\sum_{i=1}^{n} w_i$ is very small.

4.1.4 Hyperparameter Evidence

The Bayesian framework we described is conditional on the parameters in the prior distribution and the parameters in the likelihood function, which can be collected, as θ, the hyperparameter vector. The normalizing constant $\mathcal{P}(\mathcal{D})$ in (4.2), 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, 1992c).

4.2 Support Vector Regression

We now describe the optimization problem (4.10) arising from the introduction of SILF (2.31) as the loss function. In this case, the MAP estimate of the function values is the minimizer of the following problem

$$\min_{\boldsymbol{f}} S(\boldsymbol{f}) = C \sum_{i=1}^{n} \ell_{\epsilon,\beta}(y_i - f(x_i)) + \frac{1}{2} \boldsymbol{f}^T \Sigma^{-1} \boldsymbol{f} \tag{4.13}$$

As usual, by introducing two slack variables ξ_i and ξ_i^*, (4.13) can be restated as the following equivalent optimization problem, which we refer to as the *primal* problem:

$$\min_{\boldsymbol{f},\boldsymbol{\xi},\boldsymbol{\xi}^*} S(\boldsymbol{f}, \boldsymbol{\xi}, \boldsymbol{\xi}^*) = C \sum_{i=1}^{n} (\psi(\xi_i) + \psi(\xi_i^*)) + \frac{1}{2} \boldsymbol{f}^T \Sigma^{-1} \boldsymbol{f} \tag{4.14}$$

subject to

$$\begin{cases} y_i - f(x_i) \leq (1-\beta)\epsilon + \xi_i \\ f(x_i) - y_i \leq (1-\beta)\epsilon + \xi_i^* \\ \xi_i \geq 0, \xi_i^* \geq 0 \quad \forall i \end{cases} \tag{4.15}$$

where

$$\psi(\varsigma) = \begin{cases} \frac{\varsigma^2}{4\beta\epsilon} & if \quad \varsigma \in [0, 2\beta\epsilon) \\ \varsigma - \beta\epsilon & if \quad \varsigma \in [2\beta\epsilon, +\infty) \end{cases} \tag{4.16}$$

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 inequalities in (4.15). The Lagrangian for the *primal* problem is:

$$\begin{aligned} L(\boldsymbol{f}, \boldsymbol{\xi}, \boldsymbol{\xi}^*; \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*) = C\sum_{i=1}^{n}(\psi(\xi_i) + \psi(\xi_i^*)) + \frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f} - \sum_{i=1}^{n}\gamma_i\xi_i - \sum_{i=1}^{n}\gamma_i^*\xi_i^* \\ - \sum_{i=1}^{n}\alpha_i(\xi_i + (1-\beta)\epsilon - y_i + f(x_i)) - \sum_{i=1}^{n}\alpha_i^*(\xi_i^* + (1-\beta)\epsilon + y_i - f(x_i)) \end{aligned} \tag{4.17}$$

The KKT conditions for the *primal* problem require

$$f(x_i) = \sum_{j=1}^{n}(\alpha_j - \alpha_j^*)Cov(x_i, x_j) \qquad \forall i \tag{4.18}$$

$$C\frac{\partial\psi(\xi_i)}{\partial\xi_i} = \alpha_i + \gamma_i \qquad \forall i \tag{4.19}$$

$$C\frac{\partial\psi(\xi_i^*)}{\partial\xi_i^*} = \alpha_i^* + \gamma_i^* \qquad \forall i \tag{4.20}$$

Based on the definition of $\psi(\cdot)$ given by (4.16) and the constraint conditions (4.19) and (4.20), the equality constraint on Lagrange multipliers can be explicitly written as

$$\alpha_i + \gamma_i = C\frac{\xi_i}{2\beta\epsilon} \text{ for } 0 \leq \xi_i < 2\beta\epsilon \text{ and } \alpha_i + \gamma_i = C \text{ for } \xi_i \geq 2\beta\epsilon \quad \forall i \tag{4.21}$$

$$\alpha_i^* + \gamma_i^* = C\frac{\xi_i^*}{2\beta\epsilon} \text{ for } 0 \leq \xi_i^* < 2\beta\epsilon \text{ and } \alpha_i^* + \gamma_i^* = C \text{ for } \xi_i^* \geq 2\beta\epsilon \quad \forall i \tag{4.22}$$

If we collect all terms involving ξ_i in the Lagrangian (4.17), we get $T_i = C\psi(\xi_i) - (\alpha_i + \gamma_i)\xi$. Using (4.16) and (4.21) we can rewrite T_i as

$$T_i = \begin{cases} -\frac{(\alpha_i + \gamma_i)^2\beta\epsilon}{C} & if \quad 0 \leq \alpha_i + \gamma_i < C \\ -C\beta\epsilon & if \quad \alpha_i + \gamma_i = C \end{cases} \tag{4.23}$$

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 $\boldsymbol{\gamma}^*$:

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*,\boldsymbol{\gamma},\boldsymbol{\gamma}^*} S(\boldsymbol{\alpha},\boldsymbol{\alpha}^*,\boldsymbol{\gamma},\boldsymbol{\gamma}^*) &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)Cov(x_i,x_j) + \sum_{i=1}^{n}(\alpha_i-\alpha_i^*)y_i \\ &\quad -\sum_{i=1}^{n}(\alpha_i+\alpha_i^*)(1-\beta)\epsilon - \frac{\beta\epsilon}{C}\sum_{i=1}^{n}\left((\alpha_i+\gamma_i)^2+(\alpha_i^*+\gamma_i^*)^2\right)\end{aligned} \tag{4.24}$$

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 (4.24) is the only one where γ_i and γ_i^* appear, (4.24) is maximal when $\gamma_i = 0$ and $\gamma_i^* = 0$ $\forall i$. Therefore, the *dual* problem can be finally simplified as

$$\begin{aligned}\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} S(\boldsymbol{\alpha},\boldsymbol{\alpha}^*) &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)Cov(x_i,x_j) - \sum_{i=1}^{n}(\alpha_i-\alpha_i^*)y_i \\ &\quad +\sum_{i=1}^{n}(\alpha_i+\alpha_i^*)(1-\beta)\epsilon + \frac{\beta\epsilon}{C}\sum_{i=1}^{n}\left(\alpha_i^2+\alpha_i^{*2}\right)\end{aligned} \tag{4.25}$$

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

The optimal value of the primal variables $\boldsymbol{f}$ can be obtained from the solution of (4.25) as

$$\boldsymbol{f}_{\mathrm{MP}} = \Sigma \cdot (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) \tag{4.26}$$

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 (4.11), 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 (2.28) and the equality constraints (4.21) and (4.22), we notice that the noise δ_i in (4.1) 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 1 *From (2.30), 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 4.3). However, it should be pointed out that the noise values δ_i seldom fall exactly on the boundary of $\Delta_{M^*} \cup \Delta_M$, since it is of low probability for a continuous random variable to be realized on some particular values.*

[3] There is an identicalness between most probable estimate and MAP estimate in Gaussian processes.

[4] Note that the region $\Delta_{M^*} \cup \Delta_M$ is crucially determined by the parameter β in the SILF (2.28).

4.2.1 General Formulation

Like SILF, the dual problem in (4.25) is a generalization of several SVR formulations (Chu et al., 2004). More exactly, when $\beta = 0$ (4.25) becomes the SVR formulation using ϵ-ILF; when $\beta = 1$, (4.25) becomes that when the Huber's loss function is used; and when $\beta = 0$ and $\epsilon = 0$, (4.25) becomes that for the case of the Laplacian loss function. Moreover, for the case of Gaussian noise model (2.8), the *dual* problem becomes

$$\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)Cov(x_i, x_j) - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)y_i + \frac{\sigma^2}{2}\sum_{i=1}^{n}\left(\alpha_i^2 + \alpha_i^{*2}\right) \tag{4.27}$$

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 (4.25) is equivalent to the general SVR (4.25) 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 (4.27) is actually a much simpler case as

$$\min_{\boldsymbol{\nu}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\nu_i\nu_j Cov(x_i, x_j) - \sum_{i=1}^{n}\nu_i y_i + \frac{\sigma^2}{2}\sum_{i=1}^{n}\nu_i^2 \tag{4.28}$$

without any constraint. This is an unconstrained quadratic programming problem. The solution on small data sets can be simply found by doing a matrix inverse. Conjugate gradient algorithm (Fletcher, 1987) can also be used to find the solution. As for the SMO algorithm design, see the LS-SVMs discussed by Keerthi and Shevade (2003).

4.2.2 Convex Quadratic Programming

Obviously, the *dual* problem (4.25) is a convex quadratic programming problem. Traditional matrix-based quadratic programming techniques that use the "chunking" idea can be employed for solving (4.25). Popular SMO algorithms for classical SVR (Smola and Schölkopf, 1998; Shevade et al., 2000) could also be adapted for its solutions. In the following, we give some details on the SMO design in which the constraints $\alpha_i \cdot \alpha_i^* = 0$ $\forall i$ have been taken into consideration and pairs of variables (α_i, α_i^*) are selected simultaneously into the working set.

4.2.2.1 Optimality Conditions

The Lagrangian for the *dual* problem (4.25) is defined as:

$$\begin{aligned} L = \quad & \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\boldsymbol{Q}_{ij} - \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*) \\ & + \sum_{i=1}^{n}(1-\beta)\epsilon(\alpha_i - \alpha_i^*) - \sum_{i=1}^{n}\pi_i\alpha_i - \sum_{i=1}^{n}\psi_i\alpha_i^* \\ & - \sum_{i=1}^{n}\lambda_i(C - \alpha_i) - \sum_{i=1}^{n}\eta_i(C - \alpha_i^*) \end{aligned} \tag{4.29}$$

where $\boldsymbol{Q}_{ij} = \Sigma_{ij} + \delta_{ij}\frac{2\beta\epsilon}{C}$ with δ_{ij} the Kronecker delta.[5] Let us define

$$F_i = y_i - \sum_{j=1}^{n}(\alpha_j - \alpha_j^*)Q(x_i, x_j) \tag{4.30}$$

and the KKT conditions should be:

$$\begin{aligned} & \tfrac{\partial L}{\partial \alpha_i} = -F_i + (1-\beta)\epsilon - \pi_i + \lambda_i = 0 \\ & \pi_i \geq 0, \pi_i\alpha_i = 0, \lambda_i \geq 0, \lambda_i(C - \alpha_i) = 0, \forall i \\ & \tfrac{\partial L}{\partial \alpha_i^*} = F_i + (1-\beta)\epsilon - \psi_i + \eta_i = 0 \\ & \psi_i \geq 0, \psi_i\alpha_i^* = 0, \eta_i \geq 0, \eta_i(C - \alpha_i^*) = 0, \forall i \end{aligned}$$

These conditions can be simplified by considering five cases for each i:

$$\begin{aligned} & \text{Case 1}: \quad \alpha_i = \alpha_i^* = 0 \quad -(1-\beta)\epsilon \leq F_i \leq (1-\beta)\epsilon \\ & \text{Case 2}: \quad \alpha_i = C \qquad\qquad F_i \geq (1-\beta)\epsilon \\ & \text{Case 3}: \quad \alpha_i^* = C \qquad\qquad F_i \leq -(1-\beta)\epsilon \\ & \text{Case 4}: \quad 0 < \alpha_i < C \qquad\qquad F_i = (1-\beta)\epsilon \\ & \text{Case 5}: \quad 0 < \alpha_i^* < C \qquad\qquad F_i = -(1-\beta)\epsilon \end{aligned} \tag{4.31}$$

We can classify any one pair into one of the following five sets, which are defined as:

$$\begin{aligned} & I_{0a} = \{i : 0 < \alpha_i < C\} \\ & I_{0b} = \{i : 0 < \alpha_i^* < C\} \\ & I_0 = I_{0a} \cup I_{0b} \\ & I_1 = \{i : \alpha_i = \alpha_i^* = 0\} \\ & I_2 = \{i : \alpha_i^* = C\} \\ & I_3 = \{i : \alpha_i = C\} \end{aligned} \tag{4.32}$$

[5] The Kronecker delta is defined as $\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$.

Let us denote $I_{up} = I_0 \cup I_1 \cup I_3$ and $I_{low} = I_0 \cup I_1 \cup I_2$. We further define F_i^{up} on the set I_{up} as

$$F_i^{up} = \begin{cases} F_i + (1-\beta)\epsilon & \text{if} \quad i \in I_{0b} \cup I_1 \\ F_i - (1-\beta)\epsilon & \text{if} \quad i \in I_{0a} \cup I_3 \end{cases}$$

and F_i^{low} on the set I_{low} as

$$F_i^{low} = \begin{cases} F_i + (1-\beta)\epsilon & \text{if} \quad i \in I_{0b} \cup I_2 \\ F_i - (1-\beta)\epsilon & \text{if} \quad i \in I_{0a} \cup I_1 \end{cases}$$

Then the conditions in (4.31) can be simplified as

$$F_i^{up} \geq 0 \ \forall i \in I_{up} \quad \text{and} \quad F_i^{low} \leq 0 \ \forall i \in I_{low}$$

Thus the stopping condition can be compactly written as:

$$b_{up} \geq -\tau \text{ and } b_{low} \leq \tau \tag{4.33}$$

where $b_{up} = \min\{F_i^{up} : i \in I_{up}\}$, $b_{low} = \max\{F_i^{low} : i \in I_{low}\}$, and the tolerance parameter $\tau > 0$, usually 10^{-3}. If (4.33) holds, we reach a τ-optimal solution. At the optimal solution, the training samples whose index $i \in I_0$ are *off-bound* SVs, *on-bound* SVs if $i \in I_2 \cup I_3$, and non-SVs if $i \in I_1$.

4.2.2.2 Sub-optimization Problem

Following the design proposed by Keerthi et al. (2001) (see Appendix C), we employ the two-loop approach till the stopping condition is satisfied. We update two Lagrange multipliers towards the optimal values in either *Type I* or *Type II* loop every time. In the *Type II* loop we update the pair associated with b_{up} and b_{low}, while in the *Type I* loop either b_{up} or b_{low} is chosen to be updated together with the variable which violates KKT conditions. The algorithm is summarized in Table C.1 in which (4.33) should be used as the stopping condition.

Now we study the solution to the sub-optimization problem, i.e. how to update the α values of the violating pair. Suppose that the pair of the Lagrangian multipliers being updated are $\{\alpha_i, \alpha_i^*\}$ and $\{\alpha_j, \alpha_j^*\}$. The other Lagrangian multipliers are fixed during the updating. Thus, we only need to find the minimization solution to the sub-optimization problem. In comparison with the sub-optimization problem in classical SVR discussed in Appendix C.2, the only difference

lies in that the sub-optimization problem cannot be analytically solved here. We could choose Newton-Raphson formulation to update the two Lagrangian multipliers. Since there are only two variables in the sub-optimization problem, there is no need for matrix inverse. The sub-optimization problem can be stated as

$$\min_{\alpha_i,\alpha_i^*,\alpha_j,\alpha_j^*} \frac{1}{2}\sum_{i'=1}^{n}\sum_{j'=1}^{n}(\alpha_{i'}-\alpha_{i'}^*)(\alpha_{j'}-\alpha_{j'}^*)Q_{i'j'}-(\alpha_i-\alpha_i^*)y_i-(\alpha_j-\alpha_j^*)y_j+(1-\beta)\epsilon(\alpha_i+\alpha_i^*+\alpha_j+\alpha_j^*)$$

subject to $0 \le \alpha_i \le C$, $0 \le \alpha_i^* \le C$, $0 \le \alpha_j \le C$ and $0 \le \alpha_j^* \le C$, where $Q_{i'j'} = Cov(x_{i'}, x_{j'}) + \delta_{i'j'}\frac{2\beta\epsilon}{C}$ with $\delta_{i'j'}$ the Kronecker delta. We need to distinguish four different cases: $(\alpha_i, 0, \alpha_j, 0)$, $(\alpha_i, 0, 0, \alpha_j^*)$, $(0, \alpha_i^*, \alpha_j, 0)$, $(0, \alpha_i^*, 0, \alpha_j^*)$, as that in Figure C.2. It is easy to derive the unconstrained solution to the sub-optimization problem according to Newton-Raphson formulation. The unconstrained solutions are tabulated in Table 4.1 with $\rho = Q_{ii}Q_{jj} - Q_{ij}Q_{ij}$, $G_i = -F_i + (1-\beta)\epsilon$, $G_j = -F_j + (1-\beta)\epsilon$, $G_i^* = F_i + (1-\beta)\epsilon$, $G_j^* = F_j + (1-\beta)\epsilon$, where F_i is defined as in (4.30).

Table 4.1: Unconstrained solution in the four quadrants.

Quadrant	**Unconstrained Solution**	
I (α_i, α_j)	$\alpha_i^{new} = \alpha_i + (-Q_{jj}G_i + Q_{ij}G_j)/\rho$	$\alpha_j^{new} = \alpha_j + (Q_{ij}G_i - Q_{ii}G_j)/\rho$
II (α_i^*, α_j)	$\alpha_i^{new} = \alpha_i + (-Q_{jj}G_i^* - Q_{ij}G_j)/\rho$	$\alpha_j^{new} = \alpha_j + (-Q_{ij}G_i^* - Q_{ii}G_j)/\rho$
III (α_i^*, α_j^*)	$\alpha_i^{new} = \alpha_i + (-Q_{jj}G_i^* + Q_{ij}G_j^*)/\rho$	$\alpha_j^{new} = \alpha_j + (Q_{ij}G_i^* - Q_{ii}G_j^*)/\rho$
IV (α_i, α_j^*)	$\alpha_i^{new} = \alpha_i + (-Q_{jj}G_i - Q_{ij}G_j^*)/\rho$	$\alpha_j^{new} = \alpha_j + (-Q_{ij}G_i - Q_{ii}G_j^*)/\rho$

It may happen that for a fixed pair of indices (i, j) the initial chosen quadrant, say e.g. (α_i, α_j^*) is the one with optimal solution. In a particular case the other quadrants, (α_i, α_j) and even (α_i^*, α_j), have to be checked. It occurs (see Figure C.2) if one of the two variables hits the 0 boundary, and then the computation of the corresponding values for the variables with(out) asterisk according to the following table is required. Moreover, there is no need to solve the sub-optimization problem exactly that requires lot of iterations. In practice, we use Newton-Raphson formulation once only in the applicable quadrant.[6] Since the objective function of the sub-optimization problem is convex, the corresponding Hessian matrix is positive semi-definite. Consequently, the Newton's direction is a non-increasing direction. It is also possible to adjust the step length along the Newton's direction in order to ensure descent, but we have found from our numerical experiments that this was not needed.[7] In numerical experiments, we find that

[6] There is no significant reduction in computational cost when we use Newton-Raphson formulation in iterative way to find the exact solution of the sub-optimization problem.

[7] We also found that the convergence of the overall algorithm is faster when the change of α are limited at the initial stage of the algorithm. Hence, we place a limit to $\|\alpha^{new} - \alpha^{old}\|$ for the case where $\alpha^{old} = 0$.

the adapted algorithm can efficiently find the solution at nearly the same computational cost as that required by SMO in classical SVR. As for more details about implementation, refer to Appendix C.4.[8]

4.3 Model Adaptation

The hyperparameter vector θ contains the parameters in the prior distribution and the parameters in the likelihood function, i.e., $\theta = \{C, \epsilon, \kappa, \kappa_b\}$.[9] For a given set of θ, the MAP estimate of the functions can be found from the solution of the optimization problem (4.13) in Section 4.2. Based on the MAP estimate $\boldsymbol{f}_{\text{MP}}$, we show now how the optimal values of the hyperparameters are inferred.

4.3.1 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 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, 1992c). An explicit expression of the evidence $\mathcal{P}(\mathcal{D}|\theta)$ can be obtained after an integral over the f-space with a Taylor expansion at $\boldsymbol{f}_{\text{MP}}$. Gradient-based optimization methods can then be used to infer the optimal hyperparameters that maximize this evidence function, more exactly

$$\mathcal{P}(\mathcal{D}|\theta) = \int \mathcal{P}(\mathcal{D}|\boldsymbol{f}, \theta)\mathcal{P}(\boldsymbol{f}|, \theta)\, d\boldsymbol{f}. \tag{4.34}$$

Using the definitions of the prior probability (4.4) and the likelihood (4.8) with SILF (2.31), the evidence (4.34) can be written as

$$\mathcal{P}(\mathcal{D}|\theta) = \mathcal{Z}_{\boldsymbol{f}}^{-1}\mathcal{Z}_S^{-n} \int \exp\left(-S(\boldsymbol{f})\right)\, d\boldsymbol{f}. \tag{4.35}$$

[8] The source code can be accessed at http://guppy.mpe.nus.edu.sg/~chuwei/code/bisvm.zip. The routines in bismo_routine.cpp and bismo_takestep.cpp were written in ANSI C to solve the quadratic programming problem.

[9] 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 automatically tuning in the present work.

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 1, the probability that a sample exactly falls on the boundary is little. Thus it is quite all right to use the second order approximation

$$S(\boldsymbol{f}) \approx S(\boldsymbol{f}_{\mathrm{MP}}) + \frac{1}{2}(\boldsymbol{f} - \boldsymbol{f}_{\mathrm{MP}})^T \cdot \left.\frac{\partial^2 S(\boldsymbol{f})}{\partial \boldsymbol{f} \partial \boldsymbol{f}^T}\right|_{\boldsymbol{f}=\boldsymbol{f}_{\mathrm{MP}}} \cdot (\boldsymbol{f} - \boldsymbol{f}_{\mathrm{MP}}) \tag{4.36}$$

where $\frac{\partial^2 S(\boldsymbol{f})}{\partial \boldsymbol{f} \partial \boldsymbol{f}^T} = \Sigma^{-1} + C \cdot \Lambda$ 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 at $\boldsymbol{f}_{\mathrm{MP}}$, otherwise the entry is zero. Introducing (4.36) and $\mathcal{Z}_{\boldsymbol{f}}$ into (4.35), we get

$$\mathcal{P}(\mathcal{D}|\theta) = \exp\left(-S(\boldsymbol{f}_{\mathrm{MP}})\right) \cdot |\mathbf{I} + C \cdot \Sigma \cdot \Lambda|^{-\frac{1}{2}} \cdot \mathcal{Z}_S^{-n} \tag{4.37}$$

where $\mathbf{I}$ is a $n \times n$ identity matrix.[10]

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 (4.26), can be used to show that the negative log probability of data given hyperparameters is

$$-\ln \mathcal{P}(\mathcal{D}|\theta) = \frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \cdot \Sigma \cdot (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) + C \sum_{i=1}^{n} \ell_{\beta,\epsilon}(y_i - f_{\mathrm{MP}}(x_i)) + \frac{1}{2} \ln \left| \mathbf{I} + \frac{C}{2\beta\epsilon} \Sigma_{\mathrm{M}} \right| + n \ln \mathcal{Z}_S \tag{4.38}$$

where $\mathcal{Z}_S$ is defined as in (2.32), $\mathbf{I}$ is a $m \times m$ identity matrix. The evidence evaluation (4.38) is a convenient yardstick for model selection.

The expression in (4.38) 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 as the hyperparameters are changed. We assume that the set of *off-bound* SVs remains unchanged near the minimum of (4.38). In this region, the evidence is a smooth function of these hyperparameters. Gradient-based optimization methods could be used for the minimizer of (4.38). We usually collect $\{\ln C, \ln \epsilon, \ln \kappa_b, \ln \kappa\}$ as the set of variables to tune,[11]

[10] Refer to Section E.1 for more details in the derivation.

[11] This collection makes the optimization problem unconstrained.

and the derivatives of $-\ln \mathcal{P}(\mathcal{D}|\theta)$ with respect to these variables are

$$\begin{aligned}\frac{\partial -\ln \mathcal{P}(\mathcal{D}|\theta)}{\partial \ln C} &= C\sum_{i=1}^{n} \ell_{\epsilon,\beta}(y_i - f_{\mathrm{MP}}(x_i)) + \frac{1}{2}\mathrm{trace}\left[\left(\frac{2\beta\epsilon}{C}\mathbf{I} + \Sigma_{\mathrm{M}}\right)^{-1}\Sigma_{\mathrm{M}}\right] \\ &\quad -\frac{n}{\mathcal{Z}_S}\left(\sqrt{\frac{\beta\epsilon\pi}{C}}\cdot \mathrm{erf}(\sqrt{C\beta\epsilon}) + \frac{2}{C}\exp(-C\beta\epsilon)\right)\end{aligned} \tag{4.39}$$

$$\begin{aligned}\frac{\partial -\ln \mathcal{P}(\mathcal{D}|\theta)}{\partial \ln \epsilon} &= -C\left(\sum_{i\in I_0}\frac{(y_i - f_{\mathrm{MP}}(x_i))^2 - (1-\beta)^2\epsilon^2}{4\beta\epsilon} + \sum_{i\in I_2\cup I_3}\epsilon\right) \\ &\quad -\frac{1}{2}\mathrm{trace}\left[\left(\frac{2\beta\epsilon}{C}\mathbf{I} + \Sigma_{\mathrm{M}}\right)^{-1}\Sigma_{\mathrm{M}}\right] + \frac{n}{\mathcal{Z}_S}\left(\sqrt{\frac{\beta\epsilon\pi}{C}}\cdot \mathrm{erf}(\sqrt{C\beta\epsilon}) + 2(1-\beta)\epsilon\right)\end{aligned} \tag{4.40}$$

$$\frac{\partial -\ln \mathcal{P}(\mathcal{D}|\theta)}{\partial \ln \kappa'} = \frac{\kappa'}{2}\mathrm{trace}\left[\left(\frac{2\beta\epsilon}{C}\mathbf{I} + \Sigma_{\mathrm{M}}\right)^{-1}\frac{\partial \Sigma_{\mathrm{M}}}{\partial \kappa'}\right] - \frac{\kappa'}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T\frac{\partial \Sigma}{\partial \kappa'}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) \tag{4.41}$$

where $\kappa' \in \{\kappa_b, \kappa\}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$ is the optimal solution of (4.25), I_0, I_2 and I_3 are defined as in (4.32).[12]

4.3.2 Feature Selection

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 could be directly embedded into the covariance function (4.3) as follows (Williams, 1998):

$$Cov[f(x_i), f(x_j)] = Cov(x_i, x_j) = \kappa_0 \exp\left(-\frac{1}{2}\sum_{l=1}^{d}\kappa_l(x_i^l - x_j^l)^2\right) + \kappa_b \tag{4.42}$$

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 (4.41).

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 (4.38) 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.

[12] Refer to Section E.2 for more details in the derivation.

4.3.3 Discussion

In classical GPR, the inversion of the full $n \times n$ matrix Σ has to be done for hyperparameter inference, refer to (1.29) in Section 1.3.2. 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} could still be large and the computation of Σ_{M}^{-1} could become the most time-consuming step. The parameter β can control the number of *off-bound* SVs. In the numerical experiments given in Section 4.5, 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.[13] 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.

Another support for the manual setting on β comes from the asymptotic property of unbiased estimator (refer to Section 2.1.1.3). When the size of training data is large, it is alright to arbitrarily choose any unbiased noise model. The effect of β setting on the efficiency of SILF is discussed in Appendix A.

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.

Some Details in Implementation We briefly summarize the algorithm in Table 4.2. In practice, we always specify some uniform prior for the variables $\{\ln C, \ln \epsilon, \ln \kappa_b, \ln \kappa\}$, which is sufficiently broad but prevents the hyperparameters from being silly values. Moreover, we notice that the gradient with respect to $\ln C$ is usually much larger than the gradient with respect to the other variables, since the value of C is usually much greater than 1. This makes the step inferred by line-search in optimization package unbalanced for these variables, i.e. $\ln C$ is

[13] 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.

Table 4.2: The algorithm of Bayesian inference in support vector regression.

BISVR	Algorithm
Initialization	choose initial values for hyperparameter vector θ_0
	load the training data into memory, i.e. initialize data structures
	use the gradient-based optimization package to maximize the evidence
in the Package	While (exit condition has not been satisfied)
in Line Search	as required by the optimization package
at θ_i	call some routine to solve the quadratic programming
	evaluate the evidence by $-\ln \mathcal{P}(\mathcal{D}\|\theta_i)$ as in (4.38)
	compute the evidence gradient at θ_i as in (4.39)$\sim$(4.41)
	endwhile
at Optimal θ	construct optimal predictor and then make predictions
Termination	exit

going too far alone. We could use C instead of $\ln C$ as the variable in optimization package to reduce the unbalance. As the prior distribution, we can set C uniformly distributed in the region $[0.01, 1000]$. It rarely happens that the optimization package requires to evaluate the evidence and its gradient at C less than 1. C going towards infinity (greater than 1000) implies that the variance of the additive noise is very small, i.e. the training data are almost noise-free. As for other hyperparameters, we specify $[-5, -0.7]$ for $\ln \epsilon$, $[-13, 10]$ for $\ln \kappa_b$ and $[-17, 10]$ for $\ln \kappa$.

4.4 Error Bar in Prediction

In this section, we present the error bar in prediction on new data points (MacKay, 1992c; Bishop, 1995). This ability to provide the error bar 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 (4.4) have the joint multivariate Gaussian distribution,

$$\begin{bmatrix} \boldsymbol{f} \\ f(x) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma & \boldsymbol{k} \\ \boldsymbol{k}^T & Cov(x,x) \end{bmatrix} \right) \tag{4.43}$$

where $\boldsymbol{f}$ and Σ are defined as in (4.4), $\boldsymbol{k}^T = [Cov(x_1, x), Cov(x_2, x), \ldots, Cov(x_n, x)]$. The conditional distribution of $f(x)$ given $\boldsymbol{f}$ is a Gaussian,

$$\mathcal{P}(f(x)|\boldsymbol{f}) \propto \exp\left(-\frac{1}{2} \frac{(f(x) - \boldsymbol{f}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k})^2}{Cov(x,x) - \boldsymbol{k}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k}} \right) \tag{4.44}$$

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 $\boldsymbol{f}_{\mathrm{MP}}$, the mean of the predictive distribution for $f(x)$ is $\boldsymbol{f}_{\mathrm{MP}}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k}$, where

$\boldsymbol{f}_{\rm MP}^T \cdot \Sigma^{-1}$ is just the Lagrange multipliers $(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T$ in the solution of (4.25).[14]

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 $\boldsymbol{f}$.[15] 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 (4.44) and $\mathcal{P}(\boldsymbol{f}|\mathcal{D})$ is given by (4.9). We replace $\boldsymbol{f} \cdot \Sigma^{-1}$ by its linear expansion around $\boldsymbol{f}_{\rm MP}$ and use the approximation (4.36) for $S(\boldsymbol{f})$, the distribution $\mathcal{P}(f(x)|\mathcal{D})$ can be written as:

$$\begin{aligned} \mathcal{P}(f(x)|\mathcal{D}) \quad \propto \quad & \int \exp\left(-\frac{1}{2}\frac{(f(x) - \boldsymbol{f}_{\rm MP}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k} - (\boldsymbol{f} - \boldsymbol{f}_{\rm MP})^T \cdot \Sigma^{-1} \cdot \boldsymbol{k})^2}{Cov(x,x) - \boldsymbol{k}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k}}\right) \cdot \\ & \exp\left(-\frac{1}{2}(\boldsymbol{f} - \boldsymbol{f}_{\rm MP})^T(\Sigma^{-1} + C \cdot \Lambda)(\boldsymbol{f} - \boldsymbol{f}_{\rm MP})\right) d\boldsymbol{f} \end{aligned} \tag{4.45}$$

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

$$\mathcal{P}(f(x)|\mathcal{D}) = \frac{1}{\sqrt{2\pi}\sigma_t} \exp\left(-\frac{(f(x) - (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \cdot \boldsymbol{k})^2}{2\sigma_t^2}\right) \tag{4.46}$$

where $\sigma_t^2 = Cov(x,x) - \boldsymbol{k}_{\rm M}^T \cdot (\frac{2\beta\epsilon}{C}\mathbf{I} + \Sigma_{\rm M})^{-1} \cdot \boldsymbol{k}_{\rm M}$, where $\boldsymbol{k}_{\rm M}$ is a sub-vector of $\boldsymbol{k}$ obtained by keeping the entries associated with the *off-bound* SVs.[16]

The target t_x is a function of $f(x)$ and the noise δ as in (4.1), i.e. $t_x = f(x) + \delta$. As the noise is of zero mean, with variance σ_n^2 as given in (2.33), the variance of t_x is therefore $\sigma_t^2 + \sigma_n^2$.

4.5 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.[17] We also implemented standard GPR (Williams, 1998) and classical SVR (Vapnik, 1995) for comparison purpose. For GPR, evidence maximization was implemented to choose the optimal hyperpa-

[14] The zero Lagrange multipliers in the solution of (4.25) associated with non-SVs do not involve in prediction at all.

[15] In a fully 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.

[16] Refer to Section E.3 for more details in the derivation.

[17] 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.

rameters 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 (4.3) is used as the kernel function.[18] 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.[19] 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.

4.5.1 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 (2.31), using $C = 10$, $\epsilon = 0.1$ and $\beta = 0.3$.[20] From (2.33), the noise variance σ_n^2

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

[19] 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.

[20] 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 (2.31), refer to Appendix F.

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 4.3 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 upper part of Table 4.3. 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 (2.31) 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 4.3. 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 4.1. Note that it is the Huber's loss function (2.26) 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

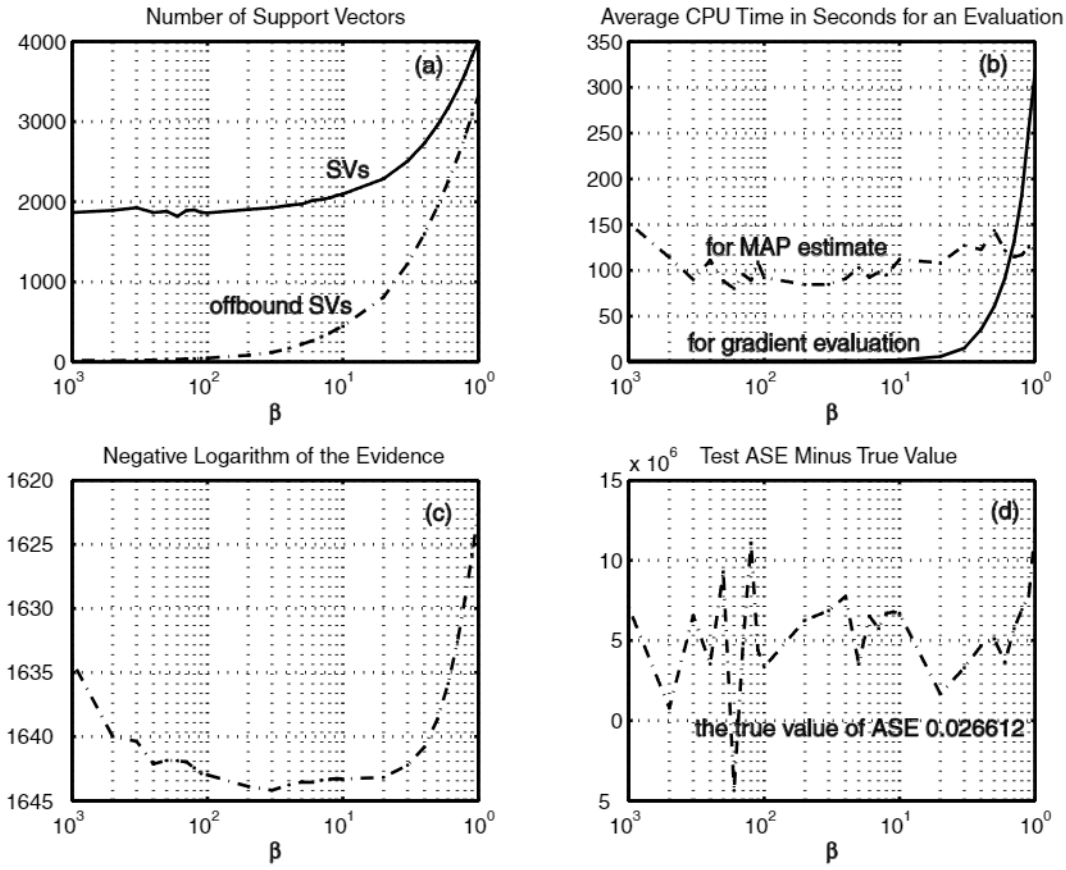

Figure 4.1: Graphs of training results with respect to different β for the 4000 sinc data set. The horizontal axis indicates the value of β in log-scale. The solid line in the graph (a) indicates the number of SVs, while the dotted line indicates the number of off-bound SVs. In the graph (b), the solid 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.

Table 4.3: 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 (2.31); $-\ln \mathcal{P}(\mathcal{D}|\theta)$ denotes the negative log evidence of the hyperparameters as in (4.38); $\mathrm{SV_M}$ 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	$-\ln \mathcal{P}_{\mathcal{D}\|\theta}$	$\mathrm{SV_M}$	$\mathrm{SV_C}$	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

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 4.2(a)∼Figure 4.2(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 4.2(g)∼Figure 4.2(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.

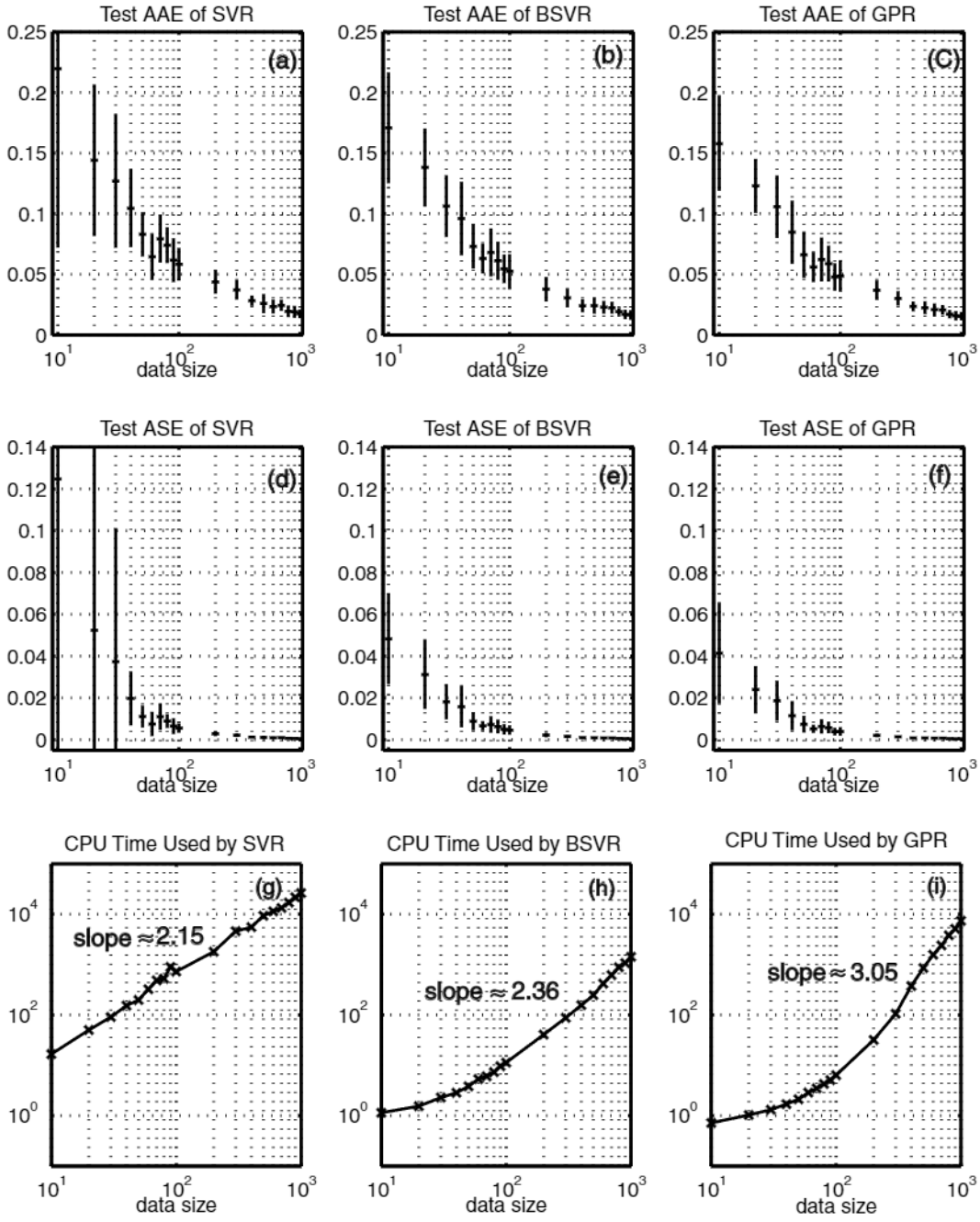

Figure 4.2: SVR, BSVR and GPR on the simulated sinc data at different training data size. The results of AAE and ASE are presented in the graph (a)∼(f) respectively. BSVR and GPR used evidence maximization to choose optimal hyperparameters, while five-fold cross validation was used for SVR. The position 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.

4.5.2 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:

$$y_1 = 2.0\cos x_1 + 1.3\cos(x_1 + x_2) \quad \text{and} \quad y_2 = 2.0\sin x_1 + 1.3\sin(x_1 + x_2) \tag{4.47}$$

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 (1992c) which contains 600 input-target pairs.[21] 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 (4.42) starting from the default settings. The results are recorded in Table 4.4.

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 (4.47), 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.[22] We normalized the input data and kept the original target values, and then trained an ARD Gaussian model (4.42) starting from the default settings. The results are recorded in Table 4.5. It is very interesting to look at the training results of the ARD parameters in the case of 6 inputs in Table 4.5. 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 4.4 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 4.4 and 4.5).

In Table 4.6, we compared the test ASE with that in other implementations, such as neural

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

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

Table 4.4: 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 4.5: 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 4.6: 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

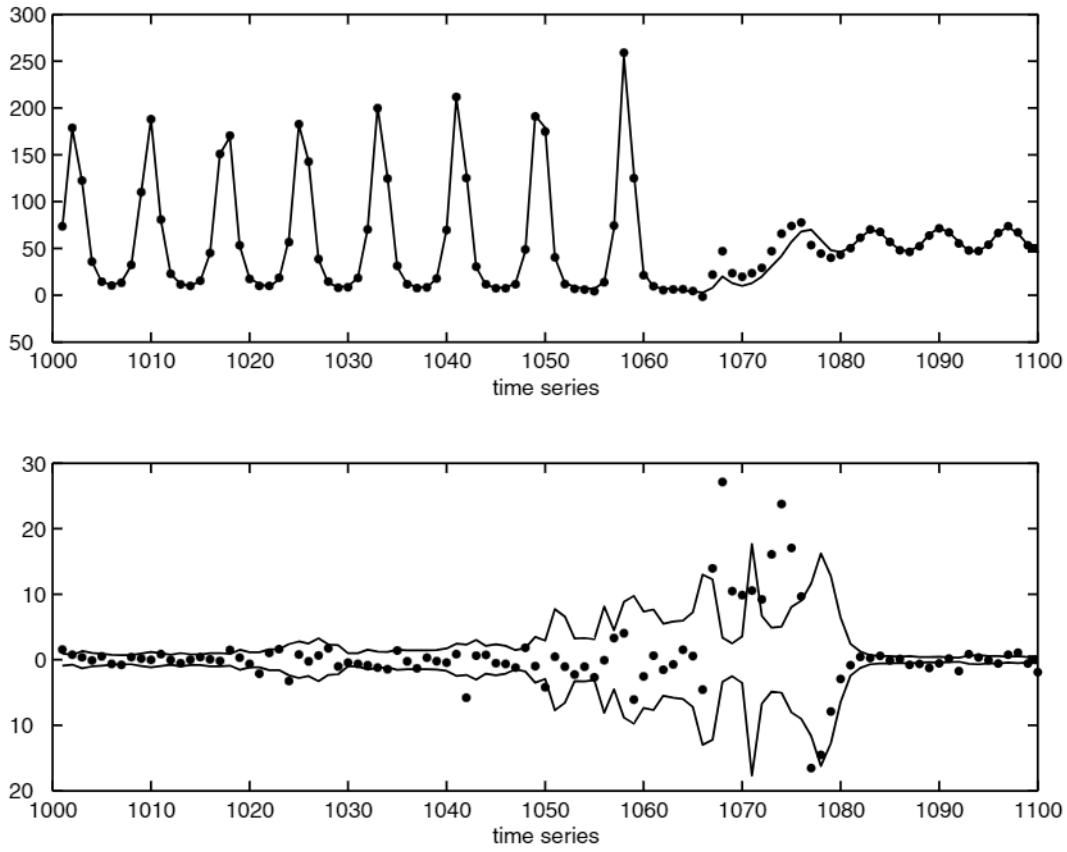

Figure 4.3: Graphs of our predictions on laser generated data. In the upper graph, the dots indicate our predictions on the testing data set and the solid curve describes the time series. In the lower graph, the dot indicates estimation error that is equal to prediction minus target, and solid curves indicate the error bars $\pm 2\sqrt{\sigma_t^2 + \sigma_n^2}$ in predictive distribution.

networks with Gaussian approximation by MacKay (1992c) 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.[23]

4.5.3 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.[24] A total of 1000 points of far-infrared

[23] 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.

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

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 (4.3) and started training from the default settings. β was fixed at 0.3. Figure 4.3 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.

4.5.4 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.[25] The objective is to predict the median house value. Following the method used by Tipping (2000) and Saunders et al. (1998),[26] the data set was partitioned into 481/25 training/testing splits randomly. This partitioning was carried out 100 times on the data. We cited the test ASE results reported by other methods in Table 4.9. The results of ARD parameters in Table 4.10 indicate that the most important attribute should be NOX, followed by TAX, LSTAT and RAD.

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.[27] 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[28] to zero mean and unit variance coordinate-wise, and then map the gender encoding (male/female/infant) into $\{(1,0,0),(0,1,0),(0,0,1)\}$.

[25] The original data can be found in StatLib, available at URL http://lib.stat.cmu.edu/datasets/boston.
[26] Saunders et al. (1998) used 80 cases in 481 training data as validation set to determine the kernel parameters.
[27] The data set and its full description can be accessed at http://www.cs.toronto.edu/~delve/data/comp-activ/.
[28] The data can be accessed via ftp://ftp.ics.uci.edu/pub/machine-learning-databases/abalone/.

Table 4.7: Training results of BSVR and standard SVR on the benchmark data sets. Both of them used the Gaussian covariance function (4.3). 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 4.8: Training results of BSVR and standard GPR on the benchmark data sets. Both of them used the ARD Gaussian covariance function (4.42). 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

The normalized data set is split into 3000 training and 1177 testing data set randomly. The 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 4.7. SVR yields significantly better performance on the computer activity data than BSVR does.[29] The results of BSVR and GPR using ARD Gaussian covariance function are presented in Table 4.8. 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

[29]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.

Table 4.9: 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 4.10: Training results of ARD hyperparameters on Boston housing data set with the fixed value of $\beta = 0.3$. The results are computed by averaging over the 100 partitions and its standard deviations are also computed.

Attribute	Description	ARD
CRIM	per capita crime rate by town	0.0363±0.0126
ZN	proportion of residential land zoned for lots over 25,000 sq. ft	0.0142±0.0043
INDUS	proportion of non-retail business acres per town	0.0659±0.0150
CHAS	Charles River dummy variable (1 if tract bounds river; 0 otherwise)	0.0329±0.0185
NOX	nitric oxides concentration (parts per 10 million)	**3.6681±1.6795**
RM	average number of rooms per dwelling	0.1415±0.0274
AGE	proportion of owner-occupied units built prior to 1940	0.0417±0.0105
DIS	weighted distances to five Boston employment centers	0.1231±0.0275
RAD	index of accessibility to radial highways	0.2400±0.0409
TAX	full-value property-tax rate per $10,000	0.9418±0.3511
PTRATIO	pupil-teacher ratio by town	0.0428±0.0083
B	$1000(Bk-0.63)^2$ where Bk is the proportion of blacks by town	0.0306±0.0135
LSTAT	% lower status of the population	0.2421±0.1120

part of the covariance matrix, the training time should be further reduced.

4.6 Summary

In this chapter, we proposed a Bayesian design for support vector regression using a unifying loss function. The SILF is smooth and also inherits most of the virtues of ϵ-ILF, such as insensitivity to outliers and sparseness in solution representation. In the Bayesian framework, we integrated support vector methods with Gaussian processes to keep the advantages of both. Various computational procedures were provided for the evaluation of MAP estimate and evidence of the hyperparameters. ARD feature selection and model adaptation were also implemented intrinsically in hyperparameter determination. Another benefit is the determination of error bar 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 showed that the generalization ability is competitive with other well-respected techniques.

Chapter 5

Extension to Binary Classification

Binary classification can be regarded as a special case of regression problems, in which the targets are only two values $\{+1, -1\}$. It is possible to take advantage of the Bayesian techniques with regression formulation to solve classification problems. For example, Van Gestel et al. (2002) employed Bayesian inference in least squares regression formulation together with a post-processer for classifier. However, such a scheme is not a direct Bayesian design for classifier, since the regression outputs have to be post-processed separately. Moreover, Bayesian inference with regression formulation to implement model selection for classifier faces the danger of over-fitting. This is because regression formulation gives punishment to any deviation from the target.[1]

The solution of the problem of classification can be considered in the light of the main principle for solving problems using a restricted amount of information formulated by Vapnik (1995, pg. 28):

> *When solving a given problem, try to avoid solving a more general problem as an intermediate step. One must try to find the desired function "directly" rather than first estimating the densities and then using the estimated densities to construct the desired function.*

Hence, any uses of regression formulation with some post-processing to solve classification problems are not desirable.

Gaussian processes provide promising non-parametric Bayesian approaches to classification problems. In standard Gaussian processes for classification (Williams and Barber, 1998), it is assumed that the likelihood of an output y (i.e. class label) for a given input $x \in \mathbb{R}^d$ can be

[1] For classifier design, only the deviation to one side of the target needs to be punished.

evaluated by $\mathcal{P}(y|f(x))$ where $f(x) : \mathbb{R}^d \to \mathbb{R}$ is a latent function which has a Gaussian prior distribution. Traditionally, the logistic function is used in likelihood evaluation for classifier designs. This results in non-Gaussian posterior distribution for the latent functions. As a popular technique, Laplacian approximation is widely used to produce the analytical formulation (MacKay, 1992a; Bishop, 1995). The important advantage of Gaussian process models over other non-Bayesian models is the explicit probabilistic formulation. This not only builds up Bayesian framework to implement hyperparameter inference but also provides us with probabilistic class prediction. Its drawback lies in the huge increase of the computation cost in matrix inverse for large training data sets.

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 (RKHS), where a linear classification is performed. The discriminant function is constructed by solving a regularized functional via convex quadratic programming (see Appendix B.1 for a quick reference). 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, 1992c) 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 (we will highlight this issue in Section 5.1), 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 chapter, we shall put forward a Bayesian design on support vector classifier in stationary Gaussian processes. We introduce a novel loss function for SVC, called the trigonometric loss function (Chu et al., 2003), 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. This differs from standard Gaussian processes for classification. 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 with the benefit of sparseness, and probabilistic class prediction can also be provided for test patterns.

This chapter is organized as follows: in section 5.1 we highlight the normalization requirement in Bayesian design for SVC; in section 5.2 we summarize the desirable characteristics of the popular loss functions for binary classification, and then propose the trigonometric loss function; in section 5.3 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 5.4 we discuss the probabilistic class prediction; we show the results of numerical experiments that verify the approach in section 5.5.

5.1 Normalization Issue in Bayesian Design for Classifier

In regression problems, the discrepancy between the target value y_x and the associated latent function f_x at the input x is used to evaluate the likelihood by a specific noise model. For binary classifier designs, we prefer to measure the probability of the class label y_x for a given latent function f_x at x as the likelihood, which is a conditional probability $\mathcal{P}(y_x|f_x)$. Here, y_x is a discrete random variable, and the sum of the probabilities for all possible cases of y_x should be equal to 1, i.e. $\sum_{y_x} \mathcal{P}(y_x|f_x) = 1$, which is referred to as *normalization requirement*. Since there are only two possible labels for any input x in binary classification problems, $y_x \in \{+1, -1\}\ \forall x$, the likelihood function $\mathcal{P}(y_x|f_x)$ for binary classifiers must satisfy

$$\mathcal{P}(y_x = +1|f_x) + \mathcal{P}(y_x = -1|f_x) = 1 \tag{5.1}$$

In the probabilistic approach for binary classification, logistic function is widely used as an approximation for the discontinuous heaviside step function in likelihood evaluation (Williams and Barber, 1998). The logistic function is defined as

$$\mathcal{P}(y_x|f_x) = \frac{1}{1 + \exp(-y_x \cdot f_x)} \tag{5.2}$$

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. Another choice is the probit function (Neal, 1997b), which is the cumulative Gaussian distribution defined as

$$\mathcal{P}(y_x|f_x) = \frac{1}{2} + \frac{1}{2}\mathrm{erf}\left(\frac{y_x \cdot f_x}{\sqrt{2}\sigma_0}\right) \tag{5.3}$$

where the noise variance $\sigma_0 > 0$ and $\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z \exp(-t^2)\,dt$. Note that both the logistic function (5.2) and the probit function (5.3) satisfy the *normalization requirement* (5.1).

The loss function associated with the shifted heaviside step function in SVC is also called hard margin loss function, which is defined as

$$\ell_h(y_x \cdot f_x) = \begin{cases} 0 & if \quad y_x \cdot f_x \geq +1; \\ +\infty & otherwise. \end{cases} \tag{5.4}$$

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

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

where ρ is a positive integer. Considering the *normalization requirement* (5.1), the corresponding likelihood function in probabilistic framework could be written as

$$\mathcal{P}(y_x|f_x) = \frac{1}{\nu(f_x)} \cdot \exp(-\ell_\rho(y_x \cdot f_x)), \tag{5.5}$$

where $y_x \in \{-1, +1\}$ and the normalizer should be

$$\nu(f_x) = \exp(-\ell_\rho(+f_x)) + \exp(-\ell_\rho(-f_x)). \tag{5.6}$$

Notice that the normalizer $\nu(f_x)$ is dependent on the latent function f_x (see Figure 5.1). As usual, we specify the prior as $\mathcal{P}(\boldsymbol{f}) \propto \exp\left(-\lambda\|\boldsymbol{f}\|^2_{\mathrm{RKHS}}\right)$ where λ is a regularization factor and $\boldsymbol{f}$ denotes the set of f_x, and use (5.5) as the likelihood function, i.e. $\mathcal{P}(\mathcal{D}|\boldsymbol{f}) = \prod_{x\in\mathcal{D}} \mathcal{P}(y_x|f_x)$ where $\mathcal{D}$ denotes the training data set. The posterior is then given as

$$\mathcal{P}(\boldsymbol{f}|\mathcal{D}) \propto \exp\left(-\lambda\|\boldsymbol{f}\|^2_{\mathrm{RKHS}} - \sum_{x\in\mathcal{D}} \ell_\rho(y_x \cdot f_x)\right) \cdot \prod_{x\in\mathcal{D}} \frac{1}{\nu(f_x)}$$

by Bayes' theorem. The MAP estimate on function values is equivalent to $\min_{\boldsymbol{f}} \mathcal{R}(\boldsymbol{f})$ where

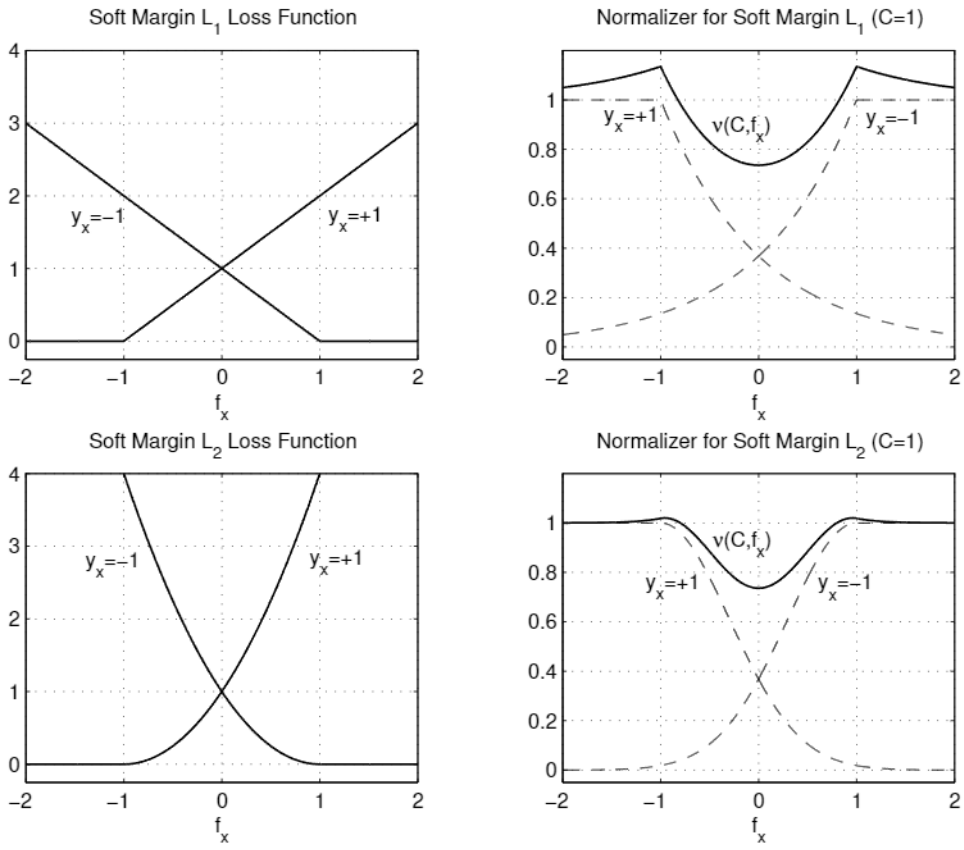

Figure 5.1: The graphs of soft margin L_1 and L_2 loss functions, together with their normalizers in likelihood. The normalizer (solid line in the two right graphs) is defined as in (5.6) with $C = 1$. The horizontal axis indicates the latent function f_x of the input vector x.

$\mathcal{R}(\boldsymbol{f}) = \lambda \|\boldsymbol{f}\|^2_{\mathrm{RKHS}} + \sum_x \ell_\rho(y_x \cdot f_x) + \sum_x \ln \nu(f_x)$. Notice that the sum of the first two terms in $\mathcal{R}(\boldsymbol{f})$ is just the objective functional in classical SVC. To compute the MAP estimate on function values f_x in this Bayesian framework, the term of the normalizer $\nu(f_x)$ has to be taken into account. This flaw precludes the solution of SVC from being directly used as the MAP estimate (Sollich, 2002), which makes us lose the computational advantage of classical SVC.

5.2 Trigonometric Loss Function

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. The 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 generalize the desirable characteristics in these loss functions 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 explicit and simple. Adhering to these requirements, we propose a novel loss function for binary classification, known as trigonometric loss function (Chu et al., 2002b). The trigonometric loss function is defined as

$$\ell_t(y_x \cdot f_x) = \begin{cases} +\infty & if \quad y_x \cdot f_x \in (-\infty, -1]; \\ 2\ln\sec(\frac{\pi}{4}(1 - y_x \cdot f_x)) & if \quad y_x \cdot f_x \in (-1, +1); \\ 0 & if \quad y_x \cdot f_x \in [+1, +\infty), \end{cases} \tag{5.7}$$

The trigonometric likelihood function is therefore written as

$$\mathcal{P}_t(y_x|f_x) = \begin{cases} 0 & if \quad y_x \cdot f_x \in (-\infty, -1]; \\ \cos^2(\frac{\pi}{4}(1 - y_x \cdot f_x)) & if \quad y_x \cdot f_x \in (-1, +1); \\ 1 & if \quad y_x \cdot f_x \in [+1, +\infty). \end{cases} \tag{5.8}$$

The derivatives of the loss function are needed in the implementation of Bayesian methods. The first order derivative of (5.7) with respect to f_x can be derived as

$$\frac{\partial \ell_t(y_x \cdot f_x)}{\partial f_x} = \begin{cases} -y_x \frac{\pi}{2}\tan(\frac{\pi}{4}(1 - y_x \cdot f_x)) & if \quad y_x \cdot f_x \in (-1, +1); \\ 0 & if \quad y_x \cdot f_x \in [+1, +\infty), \end{cases} \tag{5.9}$$

and the second order derivative is

$$\frac{\partial^2 \ell_t(y_x \cdot f_x)}{\partial f_x^2} = \begin{cases} \frac{\pi^2}{8}\sec^2(\frac{\pi}{4}(1 - y_x \cdot f_x)) & if \quad y_x \cdot f_x \in (-1, +1); \\ 0 & if \quad y_x \cdot f_x \in [+1, +\infty). \end{cases} \tag{5.10}$$

From (5.8) and Figure 5.2, it is easy to see that the normalizer $\nu(f_x)$ is a constant for any f_x. From (5.7) and Figure 5.2, 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 5.5.1, 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

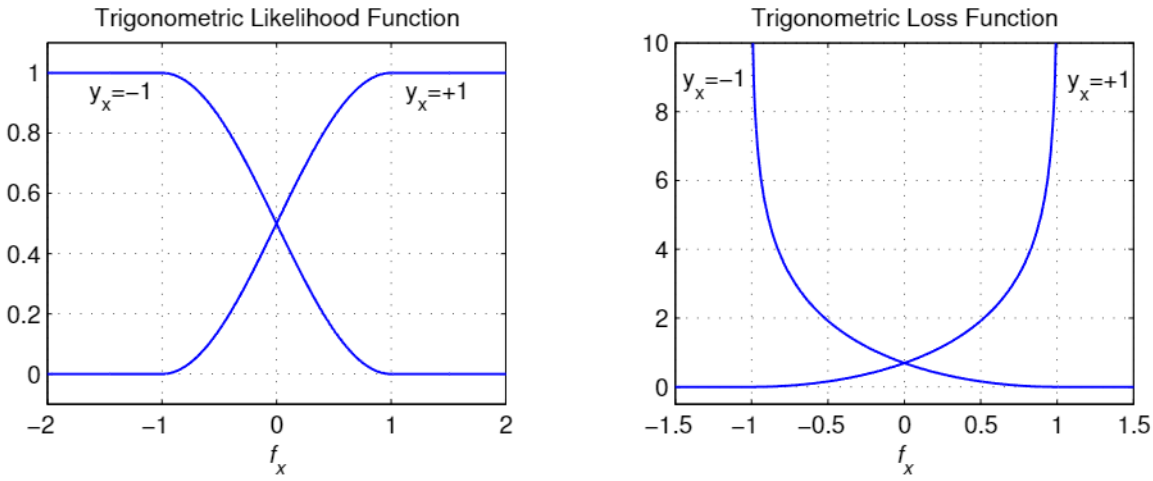

Figure 5.2: The graphs of trigonometric likelihood function and its loss function. The horizontal axis indicates the latent function f_x of the input vector x.

performance is found to be very close to that of the classical SVC method. The details are given in Section 5.5.

Remark 2 *The trigonometric loss function (5.7) could also be stated in a more general form as*

$$\ell_t(y_x \cdot f_x) = \begin{cases} +\infty & if \quad y_x \cdot f_x \in (-\infty, -\delta]; \\ 2\ln\sec\left(\frac{\pi}{4}(1 - \frac{1}{\delta} \cdot y_x \cdot f_x)\right) & if \quad y_x \cdot f_x \in (-\delta, +\delta); \\ 0 & if \quad y_x \cdot f_x \in [+\delta, +\infty), \end{cases} \tag{5.11}$$

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

5.3 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

$$\min_{f \in \mathrm{RKHS}} \mathcal{R}(f) = \sum_{i=1}^{n} \ell_t(y_{x_i} \cdot f_{x_i}) + \lambda \|f\|^2_{\mathrm{RKHS}}, \tag{5.12}$$

where the regularization parameter λ is positive and $\|f\|^2_{\mathrm{RKHS}}$ is a norm in the RKHS. As a byproduct, the TSVC along the way of classical SVC is described in the Appendix H. Here we only focus on our initial motivation to integrate with Bayesian techniques. If we assume that the prior $\mathcal{P}(f) \propto e^{-\lambda \|f\|^2_{\mathrm{RKHS}}}$ and the likelihood $\mathcal{P}(\mathcal{D}|f) \propto e^{-\sum_{i=1}^{n} \ell_t(y_{x_i} \cdot f_{x_i})}$, the minimizer of TSVC regularized functional (5.12) could be directly interpreted as maximum a posteriori

(MAP) estimate of the function $\boldsymbol{f}$ in the RKHS (Evgeniou et al., 1999). The function $\boldsymbol{f}$ could be also explained as a family of random variables in a Gaussian process due to the duality between RKHS and stochastic processes (Wahba, 1990).

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 builds 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 attempts the trigonometric loss function in place of the logistic loss function in likelihood evaluation that results in another convex programming problem in MAP estimate and sparseness in computation. This classifier, TSVC in Bayesian framework, is referred to as Bayesian TSVC (BTSVC).

5.3.1 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 could be defined as

$$Cov[f(x_i), f(x_j)] = \kappa_0 \exp\left(-\frac{1}{2}\kappa\|x_i - x_j\|^2\right) + \kappa_b, \tag{5.13}$$

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 (5.13) 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 should be

$$Cov[f(x_i), f(x_j)] = \kappa_0 K(x_i, x_j) + \kappa_b, \tag{5.14}$$

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 could also be used in the covariance function, such as

[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.

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

$$\mathcal{P}(\boldsymbol{f}\,|\theta) = \frac{1}{Z_{\boldsymbol{f}}}\exp(-\frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f}), \tag{5.15}$$

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 $Cov[f(x_i), f(x_j)]$.[3]

The likelihood with the trigonometric likelihood function (5.8) can be written as

$$\mathcal{P}(\mathcal{D}|\boldsymbol{f}, \theta) = \prod_{i=1}^{n}\mathcal{P}_t(y_{x_i}|f(x_i)). \tag{5.16}$$

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

$$\mathcal{P}(\boldsymbol{f}\,|\mathcal{D}, \theta) = \frac{1}{Z_S}\exp\left(-S(\boldsymbol{f})\right), \tag{5.17}$$

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 (5.7) 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

$$\min_{\boldsymbol{f}} S(\boldsymbol{f}) = \frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f} + \sum_{i=1}^{n}\ell_t(y_{x_i}\cdot f(x_i)). \tag{5.18}$$

This is a regularized functional. If $\boldsymbol{f}_{\mathrm{MP}}$ denotes an optimal solution of (5.18), 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 (5.9) and $\boldsymbol{\upsilon}$ as the column vector containing $\{\upsilon_i\}$. Then $\boldsymbol{f}_{\mathrm{MP}}$ can be written as:

$$\boldsymbol{f}_{\mathrm{MP}} = \Sigma\cdot\boldsymbol{\upsilon}. \tag{5.19}$$

[3] It is possible to insert "jitter" term in the diagonal entries of the covariance matrix, that could reflect the uncertainty in the corresponding function value.

Using (5.14), we can decompose the solution (5.19) into the form

$$f_{\mathrm{MP}}(x) = \sum_{i=1}^{n} \upsilon_i \cdot \kappa_0 \cdot K(x, x_i) + \kappa_b \sum_{i=1}^{n} \upsilon_i, \tag{5.20}$$

to show the significance of the hyperparameters.[4] 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 (5.20). In the case of high noise level, a smaller value κ_0 could reduce the deleterious effect from some particular outliers. In the regularized functional (5.18), 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 (5.20).[5]

5.3.2 Convex Programming

In this subsection, we formulate the optimization problem (5.18) as a convex programming problem, and then adapt 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 (5.18) can then be restated as the following equivalent optimization problem, which we refer to as the *primal* problem:

$$\min_{\boldsymbol{f},\boldsymbol{\xi}} \frac{1}{2} \boldsymbol{f}^T \Sigma^{-1} \boldsymbol{f} + 2 \sum_{i=1}^{n} \ln \sec \left(\frac{\pi}{4} \xi_i \right) \tag{5.21}$$

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 (5.21). The Lagrangian for the *primal* problem (5.21) is:

$$L(\boldsymbol{f}, \boldsymbol{\xi}) = \frac{1}{2} \boldsymbol{f}^T \Sigma^{-1} \boldsymbol{f} + 2 \sum_{i=1}^{n} \ln \sec \left(\frac{\pi}{4} \xi_i \right) - \sum_{i=1}^{n} \gamma_i \cdot \xi_i - \sum_{i=1}^{n} \alpha_i (y_{x_i} \cdot f(x_i) - 1 + \xi_i). \tag{5.22}$$

The KKT conditions for the *primal* problem (5.21) are

$$f(x_i) = \sum_{j=1}^{n} y_{x_j} \alpha_j Cov(x_i, x_j), \qquad \forall i; \tag{5.23}$$

[4] Let us consider the case that we use $K(x_i, x_j) + \kappa_b$ as covariance function (5.14) and the general trigonometric loss function (5.11) in the regularized functional (5.18). Comparing the consequent solution with that in (5.20), we can notice that there is an equivalence between κ_0 in covariance function and the parameter $1/\delta$ in (5.11).

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

$$\frac{\pi}{2}\tan\left(\frac{\pi}{4}\xi_i\right) = \alpha_i + \gamma_i, \qquad \forall i. \tag{5.24}$$

We can write (5.24) as

$$\xi_i = \frac{4}{\pi}\arctan\left(\frac{2}{\pi}(\alpha_i + \gamma_i)\right) \tag{5.25}$$

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 (5.22), we get

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

Using (5.25) we can rewrite T_i as

$$T_i = \ln\left(1 + \left(\frac{2}{\pi}(\alpha_i + \gamma_i)\right)^2\right) - \frac{4}{\pi}(\alpha_i + \gamma_i)\arctan\left(\frac{2}{\pi}(\alpha_i + \gamma_i)\right). \tag{5.26}$$

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

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\gamma}} \mathcal{R}(\boldsymbol{\alpha},\boldsymbol{\gamma}) = &-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(y_{x_i}\alpha_i)(y_{x_j}\alpha_j)Cov\left[f(x_i), f(x_j)\right] + \sum_{i=1}^{n}\alpha_i \\ &+ \sum_{i=1}^{n}\left[\ln\left(1 + \left(\frac{2}{\pi}(\alpha_i + \gamma_i)\right)^2\right) - \frac{4}{\pi}(\alpha_i + \gamma_i)\arctan\left(\frac{2}{\pi}(\alpha_i + \gamma_i)\right)\right]\end{aligned} \tag{5.27}$$

subject to

$$\alpha_i \geq 0 \text{ and } \gamma_i \geq 0, \forall i. \tag{5.28}$$

It is noted that $\mathcal{R}(\boldsymbol{\alpha},\boldsymbol{\gamma}) \leq \mathcal{R}(\boldsymbol{\alpha}, 0)$ for any $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ satisfying (5.28). Hence the maximization of (5.27) over $(\boldsymbol{\alpha},\boldsymbol{\gamma})$ satisfying (5.28) 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

$$\begin{aligned}\min_{\boldsymbol{\alpha}} \frac{1}{2}&\sum_{i=1}^{n}\sum_{j=1}^{n}(y_{x_i}\alpha_i)(y_{x_j}\alpha_j)Cov\left[f(x_i), f(x_j)\right] - \sum_{i=1}^{n}\alpha_i \\ &+ \sum_{i=1}^{n}\left[\frac{4}{\pi}\alpha_i\arctan\left(\frac{2\alpha_i}{\pi}\right) - \ln\left(1 + \left(\frac{2\alpha_i}{\pi}\right)^2\right)\right]\end{aligned} \tag{5.29}$$

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

The *dual* problem (5.29) 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 (5.29). The KKT condition for the *dual* problem (5.29) requires

$$F_i + y_{x_i}\eta_i = 0, \qquad \forall i, \tag{5.30}$$

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 (5.30) can be simplified by considering three cases for each i:

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

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

$$\beta^{up} \geq -\tau \quad \text{and} \quad \beta^{low} \leq \tau \tag{5.31}$$

where τ is a positive tolerance parameter, usually 10^{-3}. If (5.31) 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 (5.23). We write (5.23) in column vector form as

$$\boldsymbol{f}_{\text{MP}} = \Sigma \cdot \boldsymbol{\upsilon} \tag{5.32}$$

where $\boldsymbol{\upsilon} = [y_{x_1}\alpha_1, y_{x_2}\alpha_2, \ldots, y_{x_n}\alpha_n]^T$, that is consistent with the form (5.19). 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 (5.31) is satisfied. The difference is that the sub-optimization problem cannot be analytically solved. In the sub-optimization problem, we choose Newton-Raphson formula to update the two Lagrange multipliers (see Appendix G for more details). In numerical experiments, we find 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.

5.3.3 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, $\mathcal{P}(\mathcal{D}|\theta)$, known as the evidence of θ, can be used to assign a preference to alternative values of the hyperparameters θ (MacKay, 1992c). The evidence could 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.

We can get the evidence 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 (5.15) and (5.16), the evidence can also be written as

$$\mathcal{P}(\mathcal{D}|\theta) = \frac{1}{Z_{\boldsymbol{f}}} \int \exp(-S(\boldsymbol{f}))\, d\boldsymbol{f}. \tag{5.33}$$

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

$$S(\boldsymbol{f}) \approx S(\boldsymbol{f}_{\mathrm{MP}}) + \frac{1}{2}(\boldsymbol{f} - \boldsymbol{f}_{\mathrm{MP}})^T \left.\frac{\partial^2 S(\boldsymbol{f})}{\partial \boldsymbol{f} \partial \boldsymbol{f}^T}\right|_{\boldsymbol{f}=\boldsymbol{f}_{\mathrm{MP}}} (\boldsymbol{f} - \boldsymbol{f}_{\mathrm{MP}}), \tag{5.34}$$

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 (5.10). Introducing (5.34) into (5.33) 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 $\boldsymbol{f}$ (5.32) 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 3 *The negative logarithm of the evidence, which is the probability of data given hyperparameters $\mathcal{P}(\mathcal{D}|\theta)$, could be written as*

$$-\ln \mathcal{P}(\mathcal{D}|\theta) = \frac{1}{2}\upsilon_{\mathrm{M}}^T \cdot \Sigma_{\mathrm{M}} \cdot \upsilon_{\mathrm{M}} + 2\sum_{m\in \mathrm{SVs}} \ln \sec\left(\frac{\pi}{4}\xi_m\right) + \frac{1}{2}\ln|\mathbf{I} + \Sigma_{\mathrm{M}} \cdot \Lambda_{\mathrm{M}}|, \tag{5.35}$$

where $\mathbf{I}$ is the identity matrix with the size of SVs, $m \in SVs$ denotes m belongs to 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.[6] We assume that 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 (5.35) with respect to these variables are required. We give an expression of the derivatives of $-\ln \mathcal{P}(\mathcal{D}|\theta)$ with respect to these variables in the following remark.

Remark 4 *The derivatives of $-\ln \mathcal{P}(\mathcal{D}|\theta)$ with respect to the variables can be generally given as*

$$\begin{aligned}\frac{\partial -\ln \mathcal{P}(\mathcal{D}|\theta)}{\partial \ln \theta} = {} & \frac{\theta}{2} tr\left((\Lambda_{\mathrm{M}}^{-1} + \Sigma_{\mathrm{M}})^{-1}\frac{\partial \Sigma_{\mathrm{M}}}{\partial \theta}\right) - \frac{\theta}{2}\upsilon_{\mathrm{M}}^T \frac{\partial \Sigma_{\mathrm{M}}}{\partial \theta}\upsilon_{\mathrm{M}} \\ & -\frac{\theta}{2}\sum_{m\in \mathrm{SVs}} \upsilon_{\mathrm{M}}^m \left((\Lambda_{\mathrm{M}}^{-1} + \Sigma_{\mathrm{M}})^{-1} \cdot \Sigma_{\mathrm{M}}\right)_{mm} \left(\Lambda_{\mathrm{M}}^{-1}(\Lambda_{\mathrm{M}}^{-1} + \Sigma_{\mathrm{M}})^{-1}\frac{\partial \Sigma_{\mathrm{M}}}{\partial \theta}\upsilon_{\mathrm{M}}\right)^m\end{aligned} \tag{5.36}$$

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 m belongs to the index set of SVs.[7]

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 (5.36). 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.

[6] The set of points in hyperparameter space where the set of SVs changes is a set of measure zero. Therefore gradient based optimization methods applied to find the minimum of $-\ln \mathcal{P}(\mathcal{D}|\theta)$ typically do not face any numerical difficulties caused by the lack of differentiability.

[7] Refer to Section E.4 for more details in the derivation.

Remark 5 *Automatic relevance determination (ARD) could be directly embedded into the covariance function (5.13) as follows*

$$Cov[f(x_i), f(x_j)] = \kappa_0 \exp\left(-\frac{1}{2}\sum_{\iota=1}^{d} \kappa_\iota (x_i^\iota - x_j^\iota)^2\right) + \kappa_b, \tag{5.37}$$

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 $-\ln \mathcal{P}(\mathcal{D}|\theta)$ (5.35) with respect to the variables $\{\ln \kappa_\iota\}$ can be evaluated like we did in Remark 4.

5.4 Probabilistic Class Prediction

In this section, we present the probabilistic class prediction on test patterns (MacKay, 1992c; 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 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:

$$\mathcal{P}(f(x)|\boldsymbol{f}, \mathcal{D}, \theta) \propto \exp\left(-\frac{1}{2}\frac{(f(x) - \boldsymbol{f}^T\Sigma^{-1}\boldsymbol{k})^2}{Cov[f(x), f(x)] - \boldsymbol{k}^T\Sigma^{-1}\boldsymbol{k}}\right). \tag{5.38}$$

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

$$\mathcal{P}(f(x)|\mathcal{D}, \theta) = \int \mathcal{P}(f(x)|\boldsymbol{f}, \mathcal{D}, \theta)\mathcal{P}(\boldsymbol{f}\,|\mathcal{D}, \theta)\, d\boldsymbol{f}, \tag{5.39}$$

where $\mathcal{P}(\boldsymbol{f}|\mathcal{D}, \theta)$ is given as in (5.17). We make a Laplacian approximation on $S(\boldsymbol{f})$ at $\boldsymbol{f}_{\rm MP}$ as given by (5.34), and replace $\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{k}$ by its linear expansion around $\boldsymbol{f}_{\rm MP}$, i.e.,

$$\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{k} = \boldsymbol{f}_{\rm MP}^T\Sigma^{-1}\boldsymbol{k} + \boldsymbol{k}^T\Sigma^{-1}(\boldsymbol{f} - \boldsymbol{f}_{\rm MP}). \tag{5.40}$$

By computing the integral over $\boldsymbol{f}$ (5.39) with the approximation (5.34) and the linear expansion

(5.40), the distribution of $\mathcal{P}(f(x)|\mathcal{D},\theta)$ could be evaluated as a Gaussian distribution

$$\mathcal{P}(f(x)|\mathcal{D},\theta) \sim \mathcal{N}(\mu_t,\sigma_t^2) = \frac{1}{\sqrt{2\pi}\sigma_t}\exp\left(-\frac{(f(x)-\mu_t)^2}{2\sigma_t^2}\right). \tag{5.41}$$

where the mean $\mu_t = \boldsymbol{v}_{\mathrm{M}}^T\boldsymbol{k}_{\mathrm{M}}$, the variance $\sigma_t^2 = Cov[f(x),f(x)] - \boldsymbol{k}_{\mathrm{M}}^T(\Lambda_{\mathrm{M}}^{-1}+\Sigma_{\mathrm{M}})^{-1}\boldsymbol{k}_{\mathrm{M}}$,[8] and $\boldsymbol{k}_{\mathrm{M}}$ 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, 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:

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

where $\mathcal{P}(y_x|f(x),\mathcal{D},\theta)$ is evaluated by trigonometric likelihood function (5.8) and $\mathcal{P}(f(x)|\mathcal{D},\theta)$ is given by (5.41). The one dimensional integral can be easily computed as:

$$\begin{aligned}\mathcal{P}(y_x|\mathcal{D},\theta) = {} & \frac{1}{2}\mathrm{erfc}\left(\frac{1-y_x\mu_t}{\sqrt{2}\sigma_t}\right) \\ & + \int_{-1}^{+1}\cos^2\left(\frac{\pi}{4}(1-y_x f(x))\right)\mathcal{N}(\mu_t,\sigma_t^2)\,df(x)\,,\end{aligned} \tag{5.42}$$

where $\mathrm{erfc}(\nu) = \frac{2}{\sqrt{\pi}}\int_{\nu}^{+\infty}\exp(-z^2)\,dz$. The definite integral from -1 to $+1$ could 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 a monotonically increasing function of σ_t^2 when $\mu_t < 0$.

For θ in (5.42) 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 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).[9]

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

[9] 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.

Table 5.1: The optimal hyperparameters in Gaussian covariance function (5.13) 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. Evidence is indexed by $-\ln \mathcal{P}(\mathcal{D}|\theta)$ which is evaluated as in (5.35). The SVs denotes the number of SVs. The 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

5.5 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 could 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 states mentioned above to infer optimal hyperparameters. $\mathcal{N}(\mu, \sigma^2)$ is used to denote a Gaussian distribution with the mean μ and the 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.[10]

5.5.1 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 (5.13) 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, $\exp\left(-\frac{\kappa}{2}\|x_i - x_j\|^2\right)$.[11] The initial search for optimal hyperparameters was done on a 7×7 coarse grid linearly spaced in the region $\{(\log_{10} C, \log_{10} \kappa) | 0 \le \log_{10} C \le 3, -3 \le \log_{10} \kappa \le 0\}$, followed by a fine search on a 9×9 uniform grid linearly spaced by 0.1 in the $(\log_{10} C, \log_{10} \kappa)$

[10] 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.

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

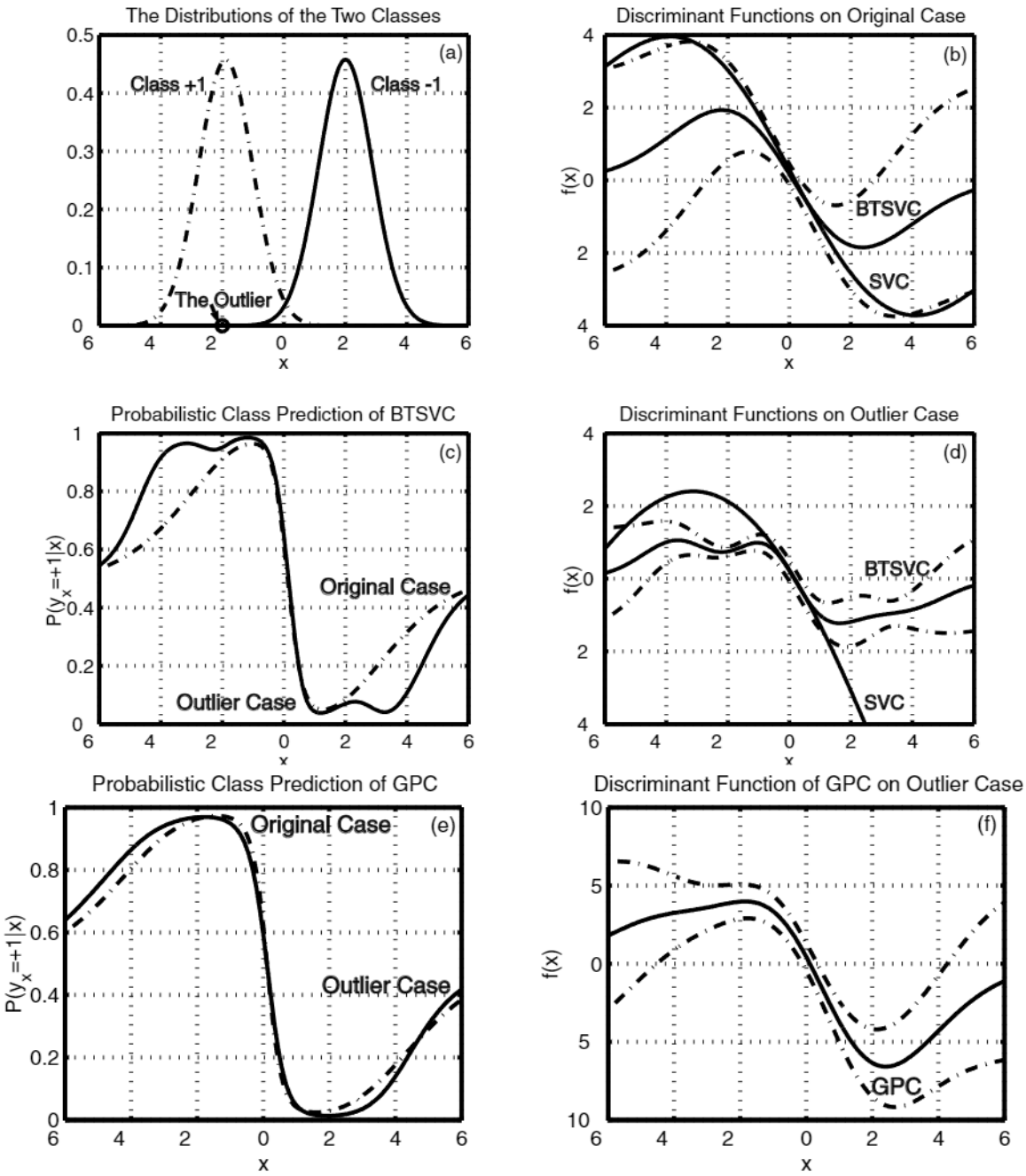

Figure 5.3: The training results of BTSVC on the one-dimensional simulated data, together with the results of SVC and GPC. In the graph (a), the distributions of each class are presented as reference. In the graph (c), we compare the probabilistic class prediction of BTSVC (5.42) on the original and outlier cases. In the graph (e), we present the results of GPC on the two cases. In the graph (b) and (d), we plot the discriminant function of BTSVC, μ_t in (5.41), 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 the graph (f), we present the results of GPC on the outlier case as reference. The dotted curves indicate the error bars provided by GPC.

Table 5.2: 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

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 5.1. Comparing the results of BTSVC on the two cases, we find that the effect of the outlier could be reduced by decreasing the hyperparameter κ_0. Moreover, the increase on κ in Gaussian kernel narrow 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 5.3(b) and 5.3(d) for the two cases. In both cases, the region $\{x : f(x) > 0\}$ is quite similar for both BTSVC and SVC. In the outlier case of BTSVC, lots of the patterns around the outlier turn out to be SVs that reduce the error bar drastically. In the probabilistic class prediction as given in Figure 5.3(c), the regions $\{x : \mathcal{P}(y_x = +1|x) > 0.5\}$ are almost same for both cases. We can also notice the effect of the error bar on the predictive probability. Note 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.

5.5.2 Simulated Data 2

We compare 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 5.2, together with the results of other probabilistic approaches that includes optimal Bayes classifier (Duda et al., 2001) using the true generic model, Bayes classifier using the generic 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).[12] BTSVC and GPC yields quite

[12]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.

similar test error as we expect since both use the Laplacian approximation in Bayesian approach and the difference 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 5.4. 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 ranges from 10 to 1000. The set of 20002 test samples is used as the common test data for all training data sets. At different size, we repeat 20 times to reduce the randomness in training data generation. If the training data size is less than 100, leave one out validation error is used to determine optimal hyperparameters for SVC, otherwise 10-fold cross validation is used. The searching method we used is same as that described in Section 5.5.1, and the test error was obtained using the optimal hyperparameters. The comparison of generalization performance and the computational cost is given in Figure 5.5. 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. From the scaling result in the three lower graphs of Figure 5.5, we find that each evaluation in GPC consumes more CPU time than BTSVC, and BTSVC consumes slightly more CPU time than SVC used for quadratic programming. However SVC with cross validation requires hundreds of evaluations (1300 times in the case of 10-fold cross validation) while BTSVC and GPC usually require 20 times only. The evaluation time of BTSVC and GPC include the cost for the gradient. For large data sets with high noise level, we cannot obtain desirable sparseness in the BTSVC solution due to the effect from outliers. As the training data size increases, the evaluation for the gradient could become a very expensive step. However, for relatively large data sets with moderate noise level, it is suitable for BTSVC to get sparseness and then fast training speed.

5.5.3 Some Benchmark Data

We also carried out Bayesian inference with Gaussian covariance function (5.13) on the benchmark data sets used by Rätsch et al. (2001).[13] We report the training results of BTSVC on these data sets in Table 5.3. The optimal hyperparameters used throughout the training on

[13]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.

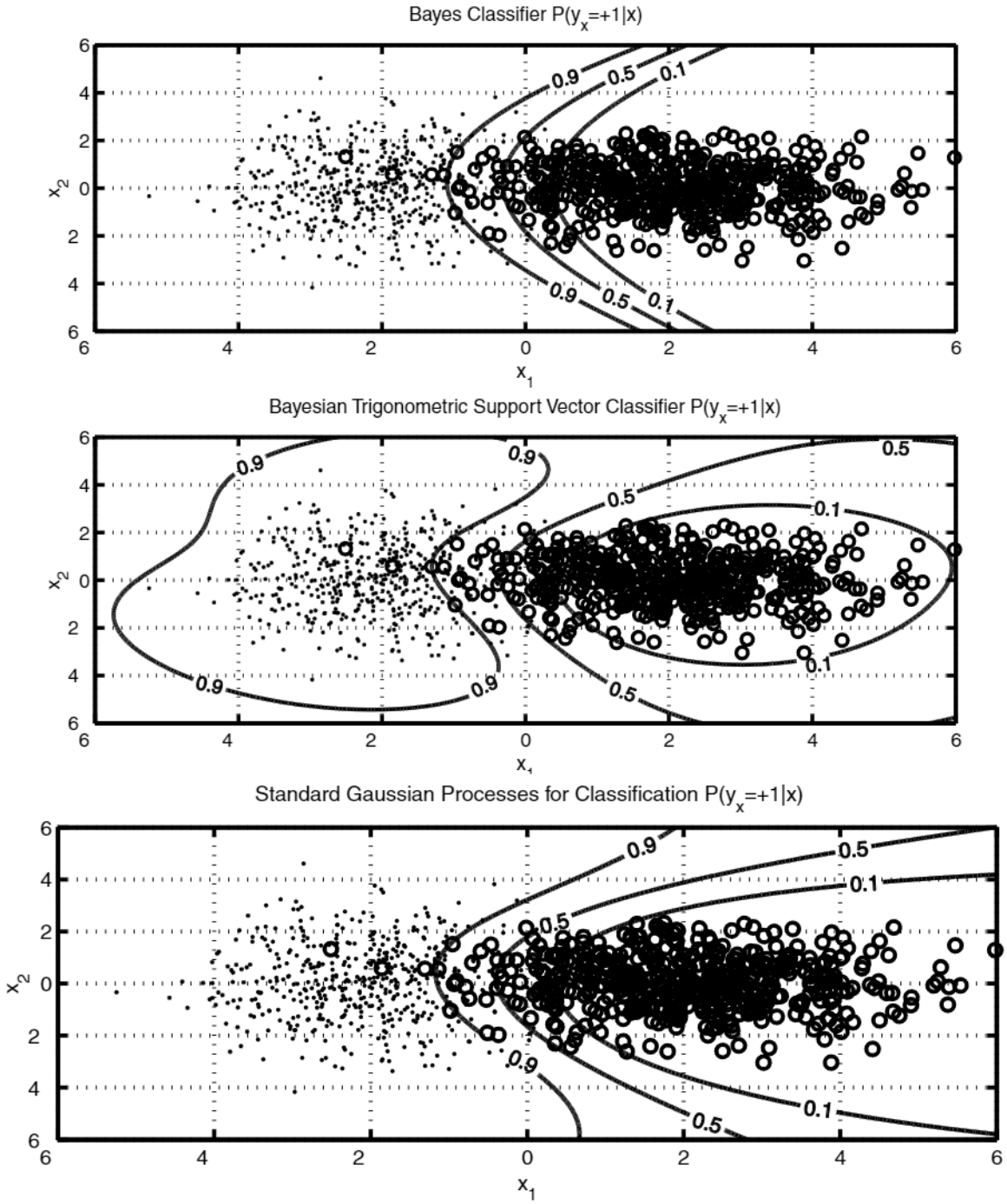

Figure 5.4: In the upper graph, the contour of the probabilistic output of Bayes classifier on the two-dimensional simulated data set is presented. In the middle graph, the contour of probabilistic output of BTSVC is presented. In the lower graph, the contour of probabilistic output of standard Gaussian processes for classification is presented. The contours are indexed by $P(y_x = +1|x, \mathcal{D}, \theta)$.

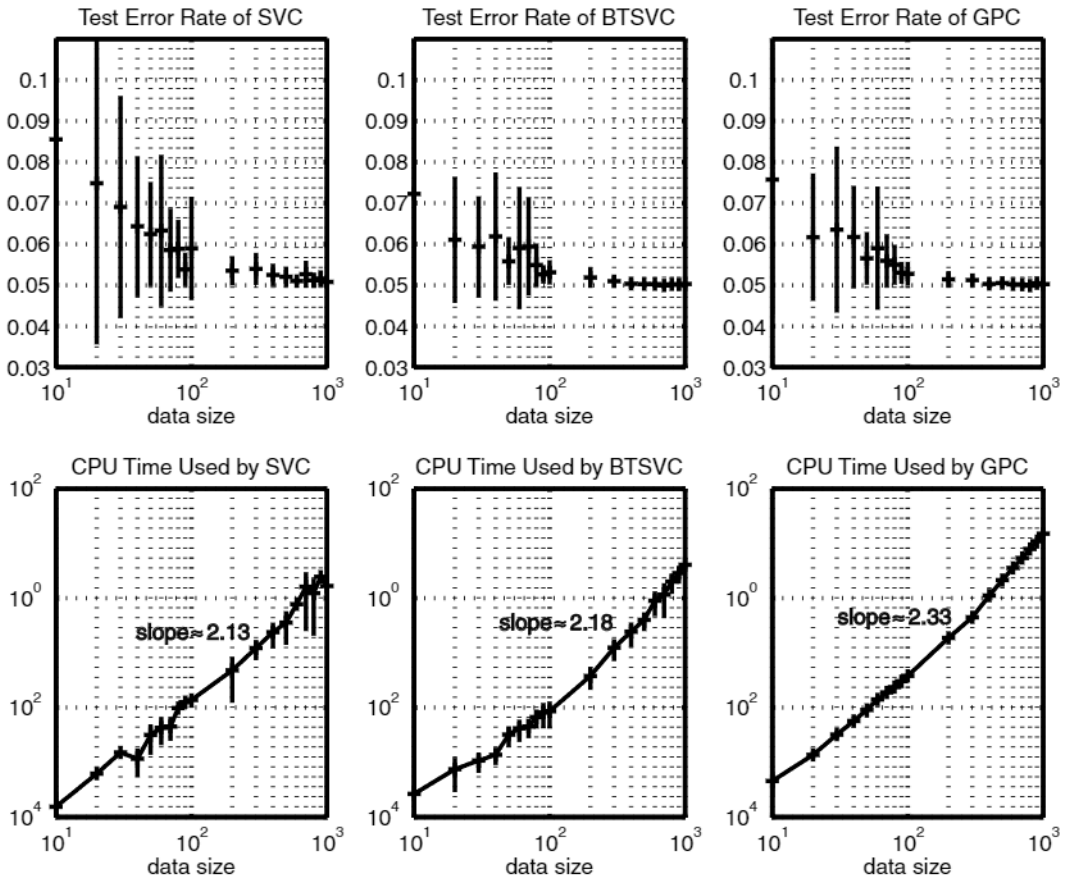

Figure 5.5: SVC, BTSVC and GPC on the two-dimensional simulated data sets at different size of training data. The test error rate at different size of training data are given in the three upper graphs separately. BTSVC and GPC used Bayesian inference with Laplacian approximation to tune hyperparameters 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 its standard deviation.

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 5.3. 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.

Table 5.3: Training results of BTSVC with Gaussian covariance function (5.13) 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 denotes the average result of BTSVC on the first five partitions. RATE denotes the test error rate in percent averaged over all partitions of that data set and the variance is also computed; 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 the bold face to indicate the cases in which the p-value satisfies the threshold 0.01.

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

We also carried out the training results of standard Gaussian processes for classification (GPC) (Williams and Barber, 1998),[14] to compare the generalization capability and computational cost with BTSVC, which can be taken as a comparison between 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 5.4. The overall generalization performance of BTSVC closely matches GPC. Notice that

[14]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 with Laplacian approximation is used to tune hyperparameters.

Table 5.4: Training results of BTSVC and GPC with Gaussian covariance function (5.13) on the 100-partition benchmark data sets. Splice* denotes training on 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 and the variance is also computed, and GPC-RATE is that of GPC. TIME denotes the average CPU time in seconds consumed by BTSVC for training on one partition, and GPC-TIME is that 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 p-value satisfies the threshold 0.01.

Data set	SVs	TIME	GPC-TIME	RATE	GPC-RATE	p-value
Banana	252.9±27.3	9.34±2.34	18.04±6.32	10.44±0.48	10.47±0.46	0.216
Breast	199.9±0.4	3.21±0.68	1.53±0.32	26.53±4.60	26.79±4.50	0.200
Diabetis	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	646.5±14.4	71.61±21.05	47.87±14.92	34.39±1.81	34.22±1.81	0.0506
German	682.7±13.8	95.71±34.76	57.78±13.93	23.48±2.11	23.81±2.17	0.0115
Heart	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	357.1±32.3	96.05±21.43	997.78±158.56	3.58±0.67	3.39±0.81	0.299
Ringnorm	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	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*	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	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	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	96.0±18.3	2.16±0.95	23.71±6.93	2.85±0.29	2.89±0.27	0.016
Waveform	190.7±22.3	4.38±1.77	30.92±6.24	10.11±0.45	10.06±0.47	0.0888

Table 5.5: Training results of BTSVC and GPC with ARD Gaussian kernel (5.37) on the Image and Splice 20-partition data sets. Splice* denotes training on 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 and the variance is also computed, and GPC-RATE is that of GPC. TIME denotes the average CPU time in seconds consumed by BTSVC for training on one partition, and GPC-TIME is that 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

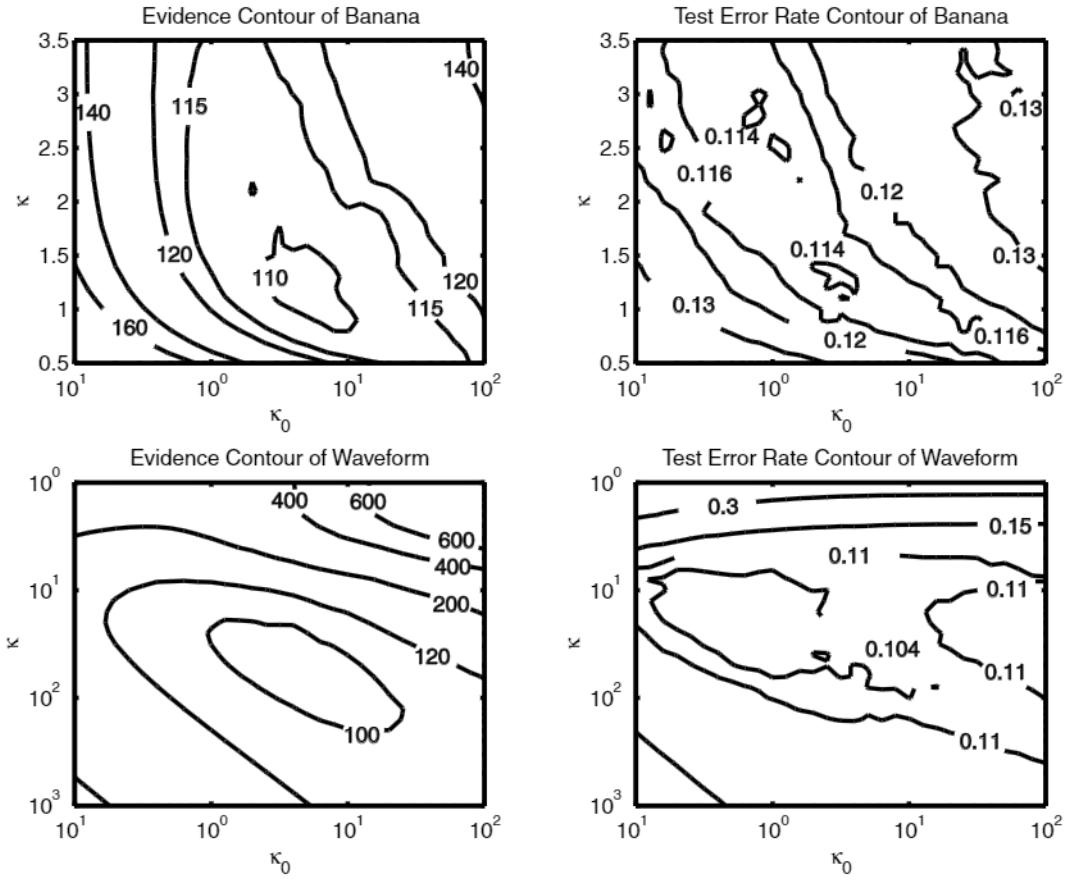

Figure 5.6: The contour graphs of evidence and testing error rate in hyperparameter space on the first fold of Banana and Waveform data sets. The horizontal axis indicates κ_0 and the vertical axis indicates κ. κ_b is fixed at 100. The evidence contour is indexed by $-\ln \mathcal{P}(\mathcal{D}|\theta)$.

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 5.6.

For the next experiment, we choose the Image and Splice data, which have many input variables, to carry out feature selection with ARD Gaussian covariance function (5.37). 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 5.5, 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 5.4) and ARD Gaussian kernel can still yields 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.

5.6 Summary

In this chapter, we proposed a Bayesian support vector classifier by introducing trigonometric likelihood function. In the probabilistic framework of stationary Gaussian processes, various computational procedures were provided for the MAP estimate and the evidence of the hyperparameters. Model adaptation and ARD feature selection were implemented intrinsically in hyperparameter inference. Furthermore, the sparseness reduces the computational cost significantly. Another benefit is the availability of class probabilities in making predictions. The results in numerical experiments verified that the generalization capability is excellent and that it is possible to tackle reasonably large data sets with moderate noise level using this approach.

Chapter 6

Conclusion

In this thesis, we developed Bayesian designs for support vector machines. In the probabilistic framework of stationary Gaussian processes along with novel loss functions, we integrated support vector methods with Gaussian processes to keep the advantages of both. Various computational procedures were provided for the MAP estimate and the evidence of the hyperparameters. Model adaptation and ARD feature selection were implemented intrinsically in hyperparameter inference. Another benefit is the availability of probabilistic evaluation in making predictions. Furthermore, the sparseness in the evidence evaluation and probabilistic prediction reduces the computational cost significantly that helps us to tackle reasonably large data sets. The results in numerical experiments indicated the usefulness of our approaches. Overall, the contributions of this work are two-fold: for classical support vector machines, we follow the standard Bayesian approach using the new loss function to implement model selection, by which it is convenient to tune hundreds of hyperparameters automatically; for standard Gaussian processes, we introduce sparseness into Bayesian computation that helps to reduce the computational burden and makes it possible to tackle large data sets of several thousands samples.

Many opportunities for future work are available within this Bayesian framework. Approximation methods other than Laplace's method for evidence evaluation are well worth extensive exploration, such as variational bounding (Seeger, 1999), mean-field statistical physics (Opper and Winther, 2000) and expectation propagation (Minka, 2001). Learning curves in Gaussian processes (Williams and Vivarelli, 2000; Opper and Vivarelli, 1999; Sollich, 1999) can be extended into our approaches to shed light on the generalization bounds. It is also straightforward to implement information-based active data selection (MacKay, 1992b) that might be helpful for large-scale learning tasks. The on-line mode of the Gaussian process learning serves as a

powerful filter. It would be quite desirable to propose some approximation propagations on the evidence that make the on-line hyperparameter adaptation possible.

References

Anderson, T. *An Introduction to Multivariate Analysis.* John Wiley, New York, 1958.

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

Bickel, P. J. and K. A. Doksum. *Mathematical Statistics - Basic Ideas and Selected Topics.* Prentice Hall, New Jersey, 1977.

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

Bochner, S. *Harmonic Analysis and the theory of probability.* Berkerley, 1979.

Boser, B., I. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifier. *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

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

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

Burtine, 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.

Chen, A. M., H. Lu, and R. Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural Computation*, 5(6):910–927, 1993.

Chu, W., S. S. Keerthi, and C. J. Ong. Extended SVMs - theory and implementation. Technical Report CD-01-14, Mechanical Engineering, National University of Singapore, 2001a.

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

Chu, W., S. S. Keerthi, and C. J. Ong. A general formulation for support vector machines. In *Proceedings of the 9th International Conference on Neural Information Processing*, pages 2522–2527, 2002a.

Chu, W., S. S. Keerthi, and C. J. Ong. A new Bayesian design method for support vector classification. In *Special session on support vector machines of the 9th International Conference on Neural Information Processing*, pages 888–892, 2002b.

Chu, W., S. S. Keerthi, and C. J. Ong. Bayesian trigonometric support vector classifier. *Neural Computation*, 15(9):2227–2254, 2003. http://guppy.mpe.nus.edu.sg/∼chuwei/paper/btsvc.pdf.

Chu, W., S. S. Keerthi, and C. J. Ong. Bayesian support vector regression using a unified loss function. *IEEE transactions on neural networks*, 15(1):29–44, 2004. http://guppy.mpe.nus.edu.sg/∼chuwei/paper/bisvr.pdf.

Collobert, R. and S. Bengio. SVMTorch: support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, Feb. 2001.

Courant, R. and D. Hilbert. *Methods of Mathematical physics*. New York, Interscience, 1953.

Cressie, N. *Statistics for Spatial Data*. Wiley, 1993.

Csató, L., E. Fokoué, M. Opper, B. Schottky, and O. Winther. Efficient approaches to Gaussian process classification. In *Advances in Neural Information Processing Systems*, volume 12, pages 251–257, 2000.

Csató, L. and M. Opper. Sparse online Gaussian processes. *Neural Computation, The MIT Press*, 14:641–668, 2002.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

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. and M. Pontil. On the V-γ dimension for regression in reproducing kernel Hilbert space. A.i. memo, Massachusette Institute of Technology, 1999.

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.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin. B*ayesian Data Analysis*. London: Chapman and Hall, 1995.

Geman, S., E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

Gibbs, M. N. *Bayesian Gaussian Processes for Regression and Classification*. Ph.D. thesis, University of Cambridge, 1997.

Girosi, F. Models of noise and robust estimates. A.I. Memo 1287, Massachusetts Institute of Technology, Artificail Intelligence Laboratory, 1991.

Girosi, F., M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(7):219–269, 1995.

Gull, S. F. Bayesian inductive inference and maximum entropy. *Maximum Entropy and Bayesian Methods in science and engineering*, 1, 1988.

Hassibi, B., D. G. Stork, and G. J. Wolff. Optimal brain surgeon and general network pruning. *IEEE International Conference on Neural Networks*, 1:293–299, 1991. San Francisco.

Haykin, S. *Neural Networks, a comprehensive foundation*. Prentice-Hall, Inc., 2 edition, 1999.

Hinton, G. E. Learning translation invariant recognition in massively parallel networks. *Proceedings PARLE Conference on Parallel Architectures and Languages Europe*, pages 1–13, 1987. Berlin:Springer-Verlag.

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

Joachims, T. Making large-sacle SVM learning practical. *Advances in Kernel Methods - Support vector learning*, pages 169–184, 1998.

Kearns, M., Y. Mansour, A. Ng, and D. Ron. An experimental and theoretical comparison of model selection methods. *Proceedings of the Eighth Annual ACM Conference on Computational Learning Theory*, 1995.

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

Keerthi, S. S. and E. G. Gilbert. Convergence of a generalized smo algorithm for svm classifier design. *Machine Learning*, 46:351–360, 2002.

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

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(3):637–649, March 2001.

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.

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

Laskov, P. An improved decomposition algorithm for regression support vector machines. *Advances in Neural Information Processing Systems 12*, pages 484–490, 2000. MIT Press.

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, 2001. Key West, Florida, USA.

LeCun, Y., J. S. Denker, and S. A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pages 598–605, 1990. San Mateo, CA: Morgan Kaufmann.

Lin, C.-J. On the convergence of the decomposition method for support vector machines. *IEEE Transactions of Neural Networks*, 12:1288–1298, 2001.

Lowe, D. G. Similarity metric learning for a variable kernel classifier. *Neural Computation*, 7: 72–85, 1995.

MacKay, D. J. C. The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736, 1992a.

MacKay, D. J. C. Information-based objective functions for active data selection. *Neural Computation*, 4(4):589–603, 1992b.

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

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.

Minka, T. P. *A family of algorithm for approximate Bayesian inference*. Ph.D. thesis, Massachusetts Institute of Technology, January 2001.

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. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

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

Neal, R. M. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report No. 9702, Department of Statistics, University of Toronto, 1997a.

Neal, R. M. Regression and classification using gaussian process priors (with discussion). In Bernerdo, J. M., J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics*, volume 6, 1997b.

O'Hagan, A. Curve fitting and optimal design for prediction (with discussion). *Journal of the Royal Statistical Society B*, 40(1):1–42, 1978.

Opper, M. and F. Vivarelli. General bounds on Bayes errors for regression with Gaussian processes. *Advances in Neural Information Processing Systems 11*, 1999.

Opper, M. and O. Winther. Gaussian processes for classification: mean field algorithm. *Neural Computation*, 12(11):2655–2684, 2000.

Papoulis, A. *Probability, Random Variables, and Stochastical Processes*. McGraw-Hill, Inc., New York, 3rd edition, 1991.

Park, J. and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3:246–257, 1991.

Parzon, E. Statistical inference on time series by RKHS methods. In *Proceedings of the 12th Biennial Seminar*, pages 1–37, 1970. Canadian Mathematical Congress, Montreal, Canada.

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.

Platt, J. C., N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In Solla, S. A., T. K. Leen, and K.-R. Mller, editors, *Advances in Neural Information Processing Systems*, pages 547–553. MIT Press, 2000.

Poggio, T. and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78:1481–1497, 1990.

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.

Press, S. J. *Bayesian Statistics: Principles, Models, and Applications*. Wiley series in probability and mathematical statistics. John Wiley and Sons, 1988.

Rasmussen, C. E. *Evaluation of Gaussian processes and other methods for non-linear regression*. Ph.D. thesis, University of Toronto, 1996.

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

Riesz, F. and B. Sz.-Nagy. *Functional Analysis*. Ungar, New York, 1955.

Saitoh, S. *Theory of Reproducing Kernels and its Applications*. Longman Scientific and Technical, Harlow, England, 1988.

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. *Learning With Kernels – Support Vector Machines, Regularization, Optimization and Beyond*. Adaptive Computation and Machine Learning. The MIT Press, December 2001.

Seeger, M. Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In *Advances in Neural Information Processing Systems*, volume 12, 1999.

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.

Small, C. G. and D. L. McLeish. *Hilbert Space Methods in Probability and Statistical Inference*. Probability and Mathematical Statisitcs. John Wiley and Sons, New York, 1994.

Smola, A. J. Regression estimation with support vector learning machines. Master's thesis, Technische Universität München, 1996.

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

Sollich, P. Learning curves for Gaussian processes. In *Advances in Neural Information Processing Systems 11*, 1999.

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

Steihaug, T. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal of Numerical Analysis*, 20(3):626–637, 1983.

Tikhonov, A. N. On solving incorrectly posed problems and methods of regularization. *Doklady Akademii Nauk USSR*, 151:501–504, 1963.

Tikhonov, A. N. and V. Y. Arsenin. *Solutions of Ill-posed Problems*. W. H. Winston, Washington, D. C., 1977.

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.

Tipping, M. E. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

Van Gestel, T., J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle. A Bayesian framework for least squares support vector machine classifiers, Gaussian processes and kernel fisher discriminant analysis. *Neural Computation*, 14:1115–1147, 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.

Vapnik, V. N. *Statistical Learning Theory*. New York: Wiley, 1998.

Vapnik, V. N. and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theoretical Probability and Its Applications*, 17:264–280, 1971.

Vapnik, V. N., S. Golowich, and A. J. Smola. Support vector method for function approximation, regression estimate, and signal processing. In *Advances in Neural Information Processing Systems 9*, pages 281–287, 1997.

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

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

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.

Williams, C. K. I. and F. Vivarelli. Upper and lower bounds on the learning curve for Gaussian processes. *Machine Learning*, 40(1):77–102, 2000.

Williams, P. M. Bayesian regularization and pruning using a Laplace prior. *Neural Computation*, 7(1):117–143, 1995.

Williamson, R. C., A. J. Smola, and B. Schölkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. Technical Report NC-TR-98-019, Royal Holloway College, University of London, UK, July 1998.

Appendix A

Efficiency of Soft Insensitive Loss Function

In classical statistics (Bickel and Doksum, 1977), there exist a Cramér and Rao lower bound for an unbiased estimator. In this section, we employ the Cramér and Rao lower bound to discuss the efficiency of the soft insensitive loss function (SILF) with the hope that it will provide us with some useful insights into complex cases in applications.

The noise density function corresponding to the SILF as in (2.28) could be stated as

$$\mathcal{P}_S(y|\mu) = \frac{1}{\mathcal{Z}_S} \begin{cases} 1 & if \quad |y-\mu| < (1-\beta)\epsilon \\ \exp\left(-C \cdot \dfrac{|y-\mu|^2}{4\beta\epsilon}\right) & if \quad (1-\beta)\epsilon \leq |y-\mu| \leq (1+\beta)\epsilon \\ \exp\big(-C \cdot (|y-\mu| - \epsilon)\big) & if \quad |y-\mu| > (1+\beta)\epsilon \end{cases} \tag{A.1}$$

where $\mathcal{Z}_S = 2(1-\beta)\epsilon + 2\sqrt{\frac{\pi\beta\epsilon}{C}} \cdot \mathrm{erf}\big(\sqrt{C\beta\epsilon}\big) + \frac{2}{C}\exp\big(-C\beta\epsilon\big)$, $\epsilon > 0$ and $0 < \beta \leq 1.0$.

Now, we consider the asymptotic estimation of a local parameter μ setting in the noise model $\mathcal{P}_S(y|\mu)$. The estimate $\hat{\mu}$ is defined as $\hat{\mu} = \arg\min_{\mu} -\ln \mathcal{P}_S(y|\mu)$. If $\ln \mathcal{P}_S(y|\mu)$ is a twice differentiable function in μ, then asymptotically, for increasing sample size to $n \to \infty$, the variance $Var_\mu\big(-\ln \mathcal{P}_S(y|\mu)\big)$ is given by, see Theorem 3.13 in Schölkopf and Smola (2001),

$$Var_\mu\big(-\ln \mathcal{P}_S(y|\mu)\big) = \frac{\displaystyle\int \left(\frac{\partial \ln \mathcal{P}_S(y|\mu)}{\partial \mu}\right)^2 d\mathcal{F}(y|\mu)}{\displaystyle\left(\int \frac{\partial^2 \ln \mathcal{P}_S(y|\mu)}{\partial \mu^2} d\mathcal{F}(y|\mu)\right)^2} \tag{A.2}$$

and the Fisher information number is well-known as

$$I(\mu) = \int \left(\frac{\partial \ln \mathcal{P}(y|\mu)}{\partial \mu} \right)^2 d\mathcal{F}(y|\mu). \tag{A.3}$$

The statistical efficiency e of an estimator $\hat{\mu}$ is defined as

$$e = \frac{1}{I(\mu) \cdot Var_{\mu}\big(-\ln \mathcal{P}_S(y|\mu) \big)} \tag{A.4}$$

Here we consider the scalar case only for simplicity. The extension to matrix is straightforward.

Following the treatment on ϵ-insensitive loss function in Section 3.4.2 of Schölkopf and Smola (2001), we discuss the efficiency of the estimator $\arg\min_{\mu} -\ln \mathcal{P}_S(y|\mu)$ on various distributions $\mathcal{F}(y|\mu)$. For this purpose, we compute the quantities $Var_{\mu}\big(-\ln \mathcal{P}_S(y|\mu) \big)$ for the noise model (A.1) as given in (A.2). We suppose that n samples of y independently drawn from the distribution $\mathcal{F}(y|\mu)$ are given. The Fisher information number in samples of size n becomes $n\,I(\mu)$. The numerator in the right hand of (A.2) could be explicitly written as:

$$\begin{aligned} &\int \left(\frac{\partial \ln \mathcal{P}_S(y|\mu)}{\partial \mu} \right)^2 d\mathcal{F}(y|\mu) = nC^2 \Bigg(\int_{\mu+(1-\beta)\epsilon}^{\mu+(1+\beta)\epsilon} \left(\frac{1}{2\beta\epsilon} \right)^2 (y-\mu)^2 \, d\mathcal{F}(y|\mu) \\ &+ \int_{\mu-(1+\beta)\epsilon}^{\mu-(1-\beta)\epsilon} \left(\frac{1}{2\beta\epsilon} \right)^2 (y-\mu)^2 \, d\mathcal{F}(y|\mu) + \int_{(1+\beta)\epsilon+\mu}^{+\infty} d\mathcal{F}(y|\mu) + \int_{-\infty}^{-(1+\beta)\epsilon+\mu} d\mathcal{F}(y|\mu) \Bigg); \end{aligned} \tag{A.5}$$

and the term in the dominator could be written as:

$$\int \frac{\partial^2 \ln \mathcal{P}_S(y|\mu)}{\partial \mu^2} \, d\mathcal{F}(y|\mu) = n \int_{\mu-(1+\beta)\epsilon}^{\mu-(1-\beta)\epsilon} \frac{C}{2\beta\epsilon} \, d\mathcal{F}(y|\mu) + n \int_{\mu+(1-\beta)\epsilon}^{\mu+(1+\beta)\epsilon} \frac{C}{2\beta\epsilon} \, d\mathcal{F}(y|\mu). \tag{A.6}$$

We may check what happened if we use the estimator for different types of noise distribution. Since Gaussian distribution is widely used in practice, we first assume that y is normally distributed with μ as mean and variance σ^2, i.e., $\mathcal{F}(y|\mu) = \int_{-\infty}^{y} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2} \right) dz$. It is easy to get that the Fisher information number is $\frac{n}{\sigma^2}$ in the n samples form (A.3). Introducing the Gaussian distribution into (A.5) and (A.6), the efficiency could be evaluated

$$e = \frac{\sigma^2 \left(erf(\frac{(1+\beta)\epsilon}{\sqrt{2}\sigma}) - erf(\frac{(1-\beta)\epsilon}{\sqrt{2}\sigma}) \right)^2}{(2\beta\epsilon)^2 (1 - erf(\frac{(1+\beta)\epsilon)}{\sqrt{2}\sigma})) + 2\int_{(1-\beta)\epsilon}^{(1+\beta)\epsilon} \frac{1}{\sqrt{2\pi}\sigma} t^2 \exp(-\frac{t^2}{2\sigma^2}) \, dt} \tag{A.7}$$

We study the relationship between the efficiency e and the parameters (β and ϵ) in the estimator. The variance σ^2 of the Gaussian distribution is set at 0.0025. We choose β as a variable and set ϵ at some fixed values. The graph of the efficiency e as a function of β is

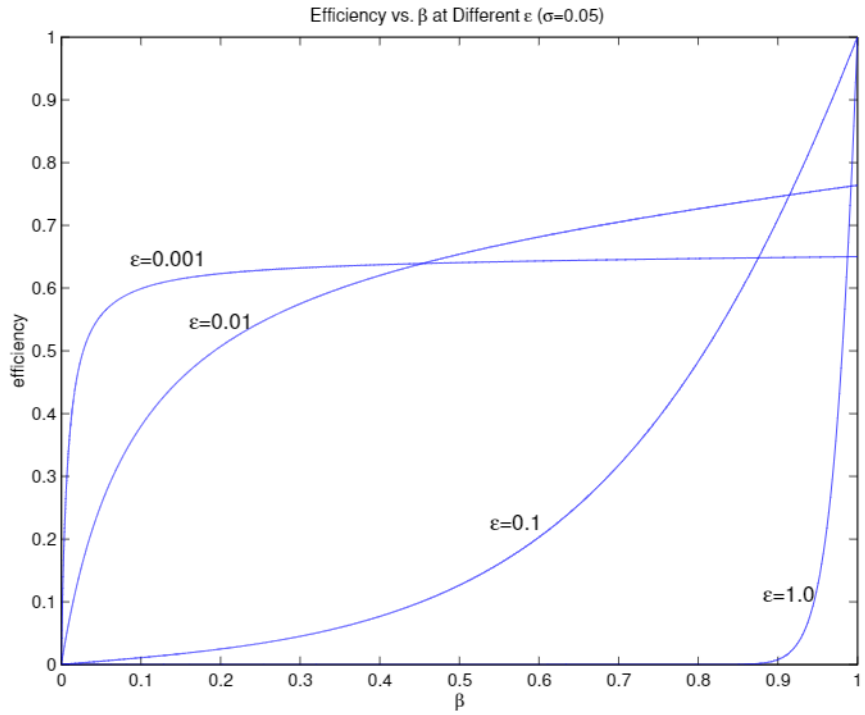

Figure A.1: Graphs of the efficiency as a function of β at different ϵ. The variance σ^2 of Gaussian noise distribution is set at 0.0025.

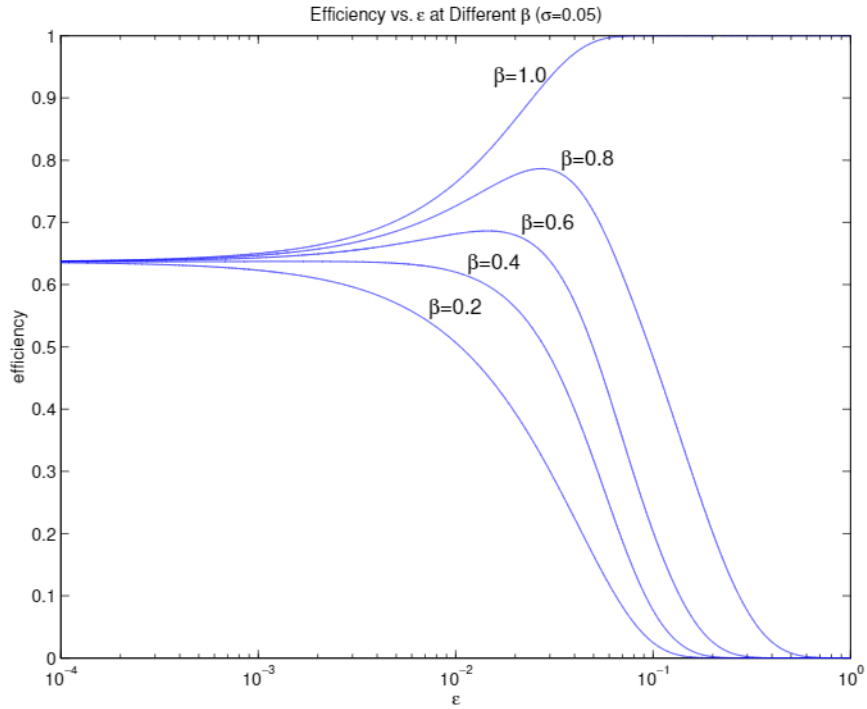

Figure A.2: Graphs of the efficiency as a function of ϵ at different β. The variance σ^2 of Gaussian noise distribution is set at 0.0025.

presented in Figure A.1. Then, we choose ϵ as a variable and set β at some fixed values. The graph of the efficiency e as a function of ϵ is presented in Figure A.2.

From Figure A.1, we find that the efficiency e is a monotonically increasing function of β, i.e., get maximal at $\beta = 1$ for any fixed values of ϵ. In other words, the Huber's loss function is best for Gaussian noise if we employ the noise model (A.1) as an estimator. From Figure A.2, we find that there is a value of ϵ that makes the efficiency e maximal for any fixed β. When β is very small, the maximal efficiency is got at Laplacian loss function, i.e., $\epsilon \to 0$. In the special case of $\beta = 1$, we find that the efficiency e approaches 1 as ϵ increases. This also verifies that the Huber's loss function with some large value of ϵ produces the same effect as the Gaussian loss function for all practical purpose.

Now we assume that y is distributed as $\mathcal{P}(y|\mu) = \frac{\lambda}{2}\exp(-\lambda|y-\mu|)$, i.e., Laplacian distribution with mean μ as the next exercise. It is easy to get that the Fisher information number is $n\,\lambda^2$ in the n samples form (A.3). Introducing the Laplacian distribution into (A.2), the variance $Var_\mu\big(-\ln \mathcal{P}_S(y|\mu)\big)$ could be evaluated as

$$e = \frac{\big(\exp(-(1-\beta)\epsilon\lambda) - \exp(-(1+\beta)\epsilon\lambda)\big)^2}{(2\beta\epsilon\lambda)^2\exp(-(1+\beta)\epsilon\lambda) + \lambda^3\int_{(1-\beta)\epsilon}^{(1+\beta)\epsilon} t^2\exp(-\lambda t)\,dt} \tag{A.8}$$

The parameter λ in Laplacian distribution is set at 5. We choose β as a variable and set ϵ at some fixed values. The graph of the efficiency e as a function of β is presented in Figure A.3. We observe that the efficiency always reaches the maximum at $\beta = 1$ for any ϵ. Then, we choose ϵ as a variable and set β at some fixed values. The graph of the efficiency e as a function of ϵ is presented in Figure A.4. We see that $\epsilon = 0$ makes the efficiency maximal for different β, which is consistent with the underlying Laplacian noise distribution.

The relationship between β and the efficiency e also gives us a hint that it is better not to regard the parameter β as free parameter in model adaptation, otherwise the noise model (A.1) is very likely to approach the Huber's loss function as $\beta \to 1$ that loses the desirable sparseness we pursue.

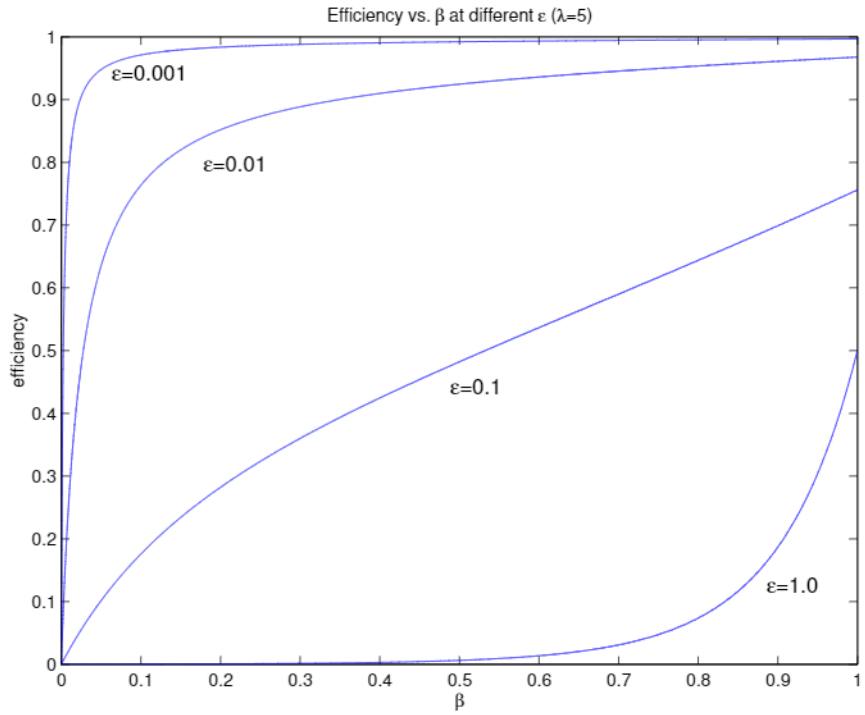

Figure A.3: Graphs of the efficiency as a function of β at different ϵ. The parameter λ of Laplacian noise distribution is set at 5.0.

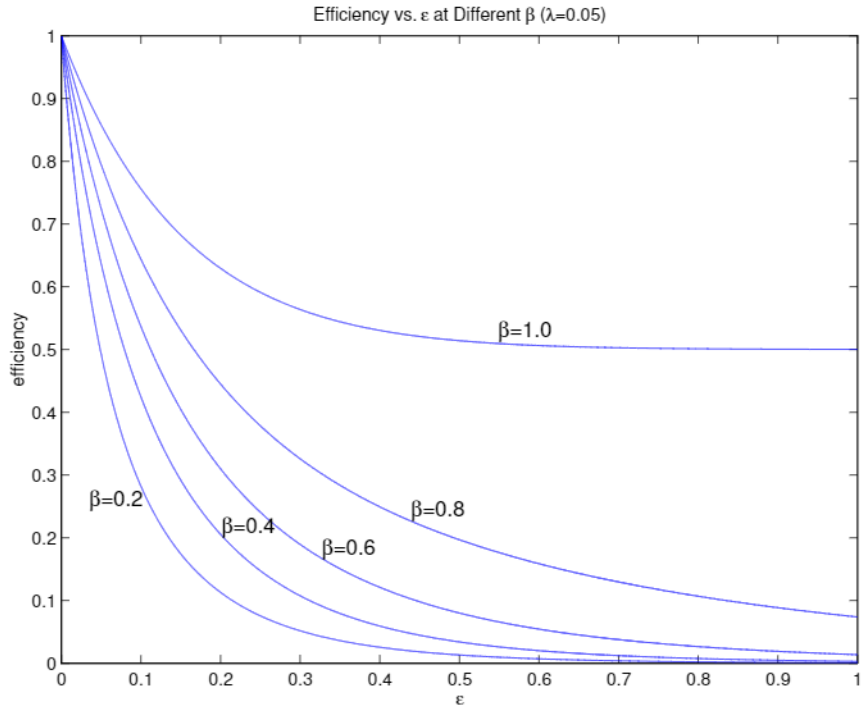

Figure A.4: Graphs of the efficiency as a function of ϵ at different β. The parameter λ of Laplacian noise distribution is set at 5.0.

Appendix B

A General Formulation of Support Vector Machines

As computationally powerful tools for supervised learning, support vector machines (SVMs) are widely used in classification and regression problems (Vapnik, 1995). Let us suppose that a data set $\mathcal{D} = \{(x_i, y_i)|i = 1, \ldots, n\}$ is given for training, where the input vector $x_i \in \mathbb{R}^d$ and y_i is the target value. SVMs maps these input vectors into a high dimensional reproducing kernel Hilbert space (RKHS), where a linear machine is constructed by minimizing a regularized functional. The linear machine takes the form of $f(x) = \langle \mathbf{w} \cdot \phi(x) \rangle + b$, where $\phi(\cdot)$ is the mapping function, b is known as the bias, and the dot product $\langle \phi(x) \cdot \phi(x') \rangle$ is also the reproducing kernel $K(x, x')$ in the RKHS. The regularized functional is usually defined as

$$\mathcal{R}(\mathbf{w}, b) = C \cdot \sum_{i=1}^{n} \ell\big(y_i, f(x_i)\big) + \frac{1}{2}\|\mathbf{w}\|^2 \tag{B.1}$$

where the regularization parameter is C which is greater than zero, the norm of $\mathbf{w}$ in the RKHS is the stabilizer and $\sum_{i=1}^{n} \ell\big(y_i, f(x_i)\big)$ is the empirical loss term. In standard SVMs, the regularized functional (B.1) can be minimized by solving a convex quadratic programming optimization problem that guarantees a unique global minimum solution.

Various loss functions can be used in SVMs that results in quadratic programming. In SVMs for classification (Burges, 1998), hard margin, L_1 soft margin and L_2 soft margin loss functions are widely used. For regression, Smola and Schölkopf (1998) have discussed a lot of common loss functions, such as Laplacian, Huber's, ϵ-insensitive and Gaussian etc.

In the following, we generalize these popular loss functions and put forward new loss functions

for classification and regression respectively. The two new loss functions are C^1 smooth that is desirable in probabilistic approaches. By introducing these loss functions in the regularized functional of classical SVMs in place of popular loss functions, we derive a general formulation for SVMs (Chu et al., 2002a). As a byproduct, the general formulation also provides a framework that facilitate the algorithm implementation.

B.1 Support Vector Classifier

In classical SVMs for binary classification (SVC) in which the target values $y_i \in \{-1, +1\}$, the hard margin loss function is defined as

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

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 classical SVC, which is defined as

$$\ell_\rho(y_x \cdot f_x) = \begin{cases} 0 & if \quad y_x \cdot f_x \geq +1; \\ \dfrac{1}{\rho}(1 - y_x \cdot f_x)^\rho & otherwise, \end{cases} \tag{B.3}$$

where ρ is a positive integer. The minimization of the regularized functional (B.1) with the soft margin (B.3) as loss function leads to a convex programming problem for any positive integer ρ; for L_1 ($\rho = 1$) or L_2 ($\rho = 2$) soft margin, it is also a convex quadratic programming problem. We generalize the L_1 and L_2 soft margin loss functions as the L_e soft margin loss function, which is defined as

$$\ell_e(y_x \cdot f_x) = \begin{cases} 0 & if \quad y_x \cdot f_x > +1; \\ \dfrac{(1 - y_x \cdot f_x)^2}{4\epsilon} & if \quad +1 \geq y_x \cdot f_x \geq +1 - 2\epsilon; \\ (1 - y_x \cdot f_x) - \epsilon & otherwise, \end{cases} \tag{B.4}$$

where the parameter $\epsilon > 0$.

From their definitions and Figure B.1, we find that the L_e soft margin approaches to L_1 soft margin as the parameter $\epsilon \to 0$. Let ϵ be fixed at some large value, the L_e soft margin approaches the L_2 soft margin for all practical purposes.

The minimization problem in SVC (B.1) with the L_e soft margin loss function can be rewritten as the following equivalent optimization problem by introducing slack variables $\xi_i \geq$

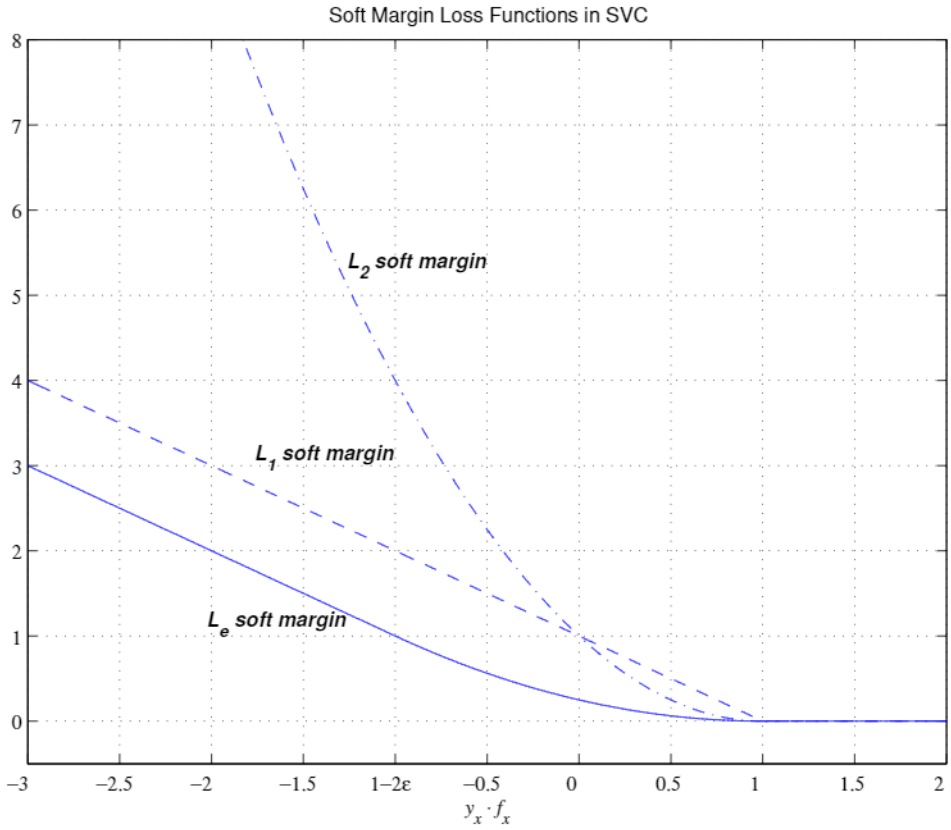

Figure B.1: Graphs of soft margin loss functions, where ϵ is set at 1.

$1 - y_i \cdot \big(\langle \mathbf{w} \cdot \phi(x_i) \rangle + b\big)\ \forall i$, which we refer to as the *primal* problem

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{R}(\mathbf{w}, b, \boldsymbol{\xi}) = C \cdot \sum_{i=1}^{n} \psi_e(\xi_i) + \frac{1}{2}\|\mathbf{w}\|^2 \tag{B.5}$$

subject to

$$\begin{cases} y_i \cdot \big(\langle \mathbf{w} \cdot \phi(x_i) \rangle + b\big) \geq 1 - \xi_i; \\ \xi_i \geq 0, \quad \forall i \end{cases} \tag{B.6}$$

where

$$\psi_e(\xi) = \begin{cases} \dfrac{\xi^2}{4\epsilon} & if \quad \xi \in [0, 2\epsilon]; \\ \xi - \epsilon & if \quad \xi \in (2\epsilon, +\infty). \end{cases} \tag{B.7}$$

Standard Lagrangian techniques (Fletcher, 1987) are used to derive the *dual* problem. Let $\alpha_i \geq 0$ and $\gamma_i \geq 0$ be the corresponding Lagrange multipliers for the inequalities in the *primal* problem (B.6), and then the Lagrangian for the *primal* problem would be:

$$L(\mathbf{w}, b, \boldsymbol{\xi}) = C \cdot \sum_{i=1}^{n} \psi_e(\xi_i) + \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \cdot \big(y_i \cdot (\langle \mathbf{w} \cdot \phi(x_i) \rangle + b) - 1 + \xi_i\big) - \sum_{i=1}^{n} \gamma_i \cdot \xi_i \tag{B.8}$$

The KKT conditions for the *primal* problem (B.5) require

$$\mathbf{w} = \sum_{i=1}^{n} y_i \cdot \alpha_i \cdot \phi(x_i) \tag{B.9}$$

$$\sum_{i=1}^{n} y_i \cdot \alpha_i = 0 \tag{B.10}$$

$$C \cdot \frac{\partial \psi_e(\xi_i)}{\partial \xi_i} = \alpha_i + \gamma_i \quad \forall i \tag{B.11}$$

Based on the definition of $\psi_e(\cdot)$ given in (B.7) and the constraint condition (B.11), an equality constraint on Lagrange multipliers can be explicitly written as

$$C \cdot \frac{\xi_i}{2\epsilon} = \alpha_i + \gamma_i \quad if \quad 0 \leq \xi_i \leq 2\epsilon \quad \text{and} \quad C = \alpha_i + \gamma_i \quad if \quad \xi_i > 2\epsilon \quad \forall i \tag{B.12}$$

If we collect all terms involving ξ_i in the Lagrangian (B.8), and let $T_i = C\psi_e(\xi_i) - (\alpha_i + \gamma_i)\xi_i$. Using (B.7) and (B.12) we have

$$T_i = \begin{cases} -\dfrac{\epsilon}{C}(\alpha_i + \gamma_i)^2 & if \quad \xi \in [0, 2\epsilon]; \\ -C\epsilon & if \quad \xi \in (2\epsilon, +\infty). \end{cases} \tag{B.13}$$

Thus the ξ_i can be eliminated if we set $T_i = -\dfrac{\epsilon}{C}(\alpha_i + \gamma_i)^2$ and introduce the additional constraints $0 \leq \alpha_i + \gamma_i \leq C$. Then the *dual* problem can be stated as a maximization problem in terms of the positive dual variables α_i and γ_i:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\gamma}} \mathcal{R}(\boldsymbol{\alpha}, \boldsymbol{\gamma}) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{i=1}^{n} \alpha_i \cdot y_i \cdot \alpha_j \cdot y_j \cdot \langle \phi(x_i) \cdot \phi(x_j) \rangle + \sum_{i=1}^{n} \alpha_i - \frac{\epsilon}{C} \sum_{i=1}^{n} (\alpha_i + \gamma_i)^2 \tag{B.14}$$

subject to

$$\alpha_i \geq 0, \gamma_i \geq 0, 0 \leq \alpha_i + \gamma_i \leq C, \forall i \text{ and } \sum_{i=1}^{n} \alpha_i \cdot y_i = 0. \tag{B.15}$$

It is noted that $\mathcal{R}(\boldsymbol{\alpha}, \boldsymbol{\gamma}) \leq \mathcal{R}(\boldsymbol{\alpha}, 0)$ for any $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ satisfying (B.15). Hence the maximization of (B.14) over $(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ can be found as maximizing $\mathcal{R}(\boldsymbol{\alpha}, 0)$ over $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{n} \alpha_i \cdot y_i = 0$. Therefore, the *dual* problem can be finally simplified as

$$\min_{\boldsymbol{\alpha}} \mathcal{R}(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{i=1}^{n} \sum_{i=1}^{n} \alpha_i \cdot y_i \cdot \alpha_j \cdot y_j \cdot K(x_i, x_j) - \sum_{i=1}^{n} \alpha_i + \frac{\epsilon}{C} \sum_{i=1}^{n} \alpha_i^2 \tag{B.16}$$

subject to $0 \leq \alpha_i \leq C, \forall i$ and $\sum_{i=1}^{n} \alpha_i \cdot y_i = 0$. With the equality (B.9), the linear classifier can be obtained from the solution of (B.16) as $f(x) = \sum_{i=1}^{n} \alpha_i \cdot y_i \cdot K(x_i, x) + b$ where b can be

easily obtained as a byproduct in the solution. In most of the cases, only some of the Lagrange multipliers, α_i, differ from zero at the optimal solution. They define the support vectors (SVs) of the problem. More exactly, the training samples (x_i, y_i) associated with α_i satisfying $0 < \alpha_i < C$ are called off-bound SVs, the samples with $\alpha_i = C$ are called on-bound SVs, and the samples with $\alpha_i = 0$ are called non-SVs.

The above formulation (B.16) is a general framework for classical SVC. There are three special cases of the formulation:

1. L_1 soft margin: the formulation is just the SVC using L_1 soft margin if we set $\epsilon = 0$.

2. Hard margin: when we set $\epsilon = 0$ and keep C large enough to prevent any α_i from reaching the upper bound C, the solution of this formulation is identical to the standard SVC with hard margin loss function.

3. L_2 soft margin: when we set $\frac{C}{2\epsilon}$ equal to the regularization parameter in SVC with L_2 soft margin, and keep C large enough to prevent any α_i from reaching the upper bound at the optimal solution, the solution will be same as that of the standard SVC with L_2 soft margin loss function.

In practice, such as on unbalanced data sets, we would like to use different regularization parameter C_+ and C_- for the samples with positive label and negative label separately.[1] As an extremely general case, we can use different regularization parameter C_i for every sample (x_i, y_i). It is straightforward to obtain the general *dual* problem as

$$\min_{\boldsymbol{\alpha}} \mathcal{R}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n}\alpha_i \cdot y_i \cdot \alpha_j \cdot y_j \cdot K(x_i, x_j) - \sum_{i=1}^{n}\alpha_i + \frac{\epsilon}{C_i}\sum_{i=1}^{n}\alpha_i^2 \tag{B.17}$$

subject to $0 \leq \alpha_i \leq C_i, \forall i$ and $\sum_{i=1}^{n}\alpha_i \cdot y_i = 0$. Obviously, the *dual* problems (B.17) is a constrained convex quadratic programming problem. Denoting $\hat{\boldsymbol{\alpha}} = [y_1\alpha_1, y_2\alpha_2, \ldots, y_n\alpha_n]$, $\boldsymbol{P} = [-y_1, -y_2, \ldots, -y_n]^T$ and $\boldsymbol{Q} = \boldsymbol{K} + \boldsymbol{\Lambda}$ where $\boldsymbol{\Lambda}$ is a $n \times n$ diagonal matrix with $\boldsymbol{\Lambda}_{ii} = \frac{2\epsilon}{C_i}$ and $\boldsymbol{K}$ is the kernel matrix with $\boldsymbol{K}_{ij} = K(x_i, x_j)$, (B.17) can be written in a general form as

$$\min_{\hat{\boldsymbol{\alpha}}} \frac{1}{2}\hat{\boldsymbol{\alpha}}^T\boldsymbol{Q}\hat{\boldsymbol{\alpha}} + \boldsymbol{P}^T\hat{\boldsymbol{\alpha}} \tag{B.18}$$

subject to $l_i \leq \hat{\alpha}_i \leq u_i$, $\forall i$ and $\sum_{i=1}^{n}\hat{\alpha}_i = 0$ where $l_i = 0$, $u_i = C_i$ when $y_i = +1$ and $l_i = -C_i$, $u_i = 0$ when $y_i = -1$. As to the algorithm design for the solution, matrix-based quadratic

[1] The ratio between C_+ and C_- is usually fixed at $\frac{C_+}{C_-} = \frac{N_-}{N_+}$, where N_+ is the number of samples with positive label and N_- is the number of samples with negative label.

programming techniques that use the "chunking" idea can be employed here. Popular SMO algorithms (Platt, 1999; Keerthi et al., 2001) could be easily adapted for the solution. For a program design and source code, refer to Chu et al. (2001a).

B.2 Support Vector Regression

Smola (1996) explained a lot of loss functions for support vector regression (SVR) that leads to a general optimization problem. There are four popular loss functions widely used for regression problems. They are

1. Laplacian loss function: $\ell_l(\delta) = |\delta|$.
2. Huber's loss function: $\ell_h(\delta) = \begin{cases} \frac{\delta^2}{4\epsilon} & if \ \ |\delta| \le 2\epsilon \\ |\delta| - \epsilon & otherwise. \end{cases}$
3. ϵ-insensitive loss function: $\ell_\epsilon(\delta) = \begin{cases} 0 & if \ \ |\delta| \le \epsilon \\ |\delta| - \epsilon & otherwise. \end{cases}$
4. Gaussian loss function: $\ell_g(\delta) = \delta^2$.

Here, we introduce another loss function to generalize these popular loss functions. The loss function known as soft insensitive loss function (SILF) is defined as:

$$\ell_{\epsilon,\beta}(\delta) = \begin{cases} |\delta| - \epsilon & if \quad |\delta| > (1+\beta)\epsilon \\ \frac{(|\delta| - (1-\beta)\epsilon)^2}{4\beta\epsilon} & if \quad (1+\beta)\epsilon \ge |\delta| \ge (1-\beta)\epsilon \\ 0 & if \quad |\delta| < (1-\beta)\epsilon \end{cases} \tag{B.19}$$

where $0 < \beta \le 1$ and $\epsilon > 0$. There is a profile of SILF as shown in Figure 2.4. The properties of SILF are entirely controlled by two parameters, β and ϵ. For a fixed ϵ, SILF approaches the ϵ-ILF as $\beta \to 0$; on the other hand, as $\beta \to 1$, it approaches the Huber's loss function. In addition, SILF becomes the Laplacian loss function as $\epsilon \to 0$. Hold ϵ fixed at some large value and let $\beta \to 1$, the SILF approach the quadratic loss function for all practical purposes. The application of SILF in Bayesian SVR has been discussed in Chu et al. (2001b).

We introduce SILF into the regularized functional (B.1) that will leads to a quadratic programming problem that could work as a general framework. As usual, two slack variables ξ_i and ξ_i^* are introduced as $\xi_i \ge y_i - \langle \mathbf{w} \cdot \phi(x_i) \rangle - b - (1-\beta)\epsilon$ and $\xi_i^* \ge \langle \mathbf{w} \cdot \phi(x_i) \rangle + b - y_i - (1-\beta)\epsilon \ \forall i$. The minimization of the regularized functional (B.1) with SILF as loss function could be rewritten

as the following equivalent optimization problem, which is usually called *primal* problem:

$$\min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \mathcal{R}(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*) = C\sum_{i=1}^{n}\left(\psi_s(\xi_i)+\psi_s(\xi_i^*)\right)+\frac{1}{2}\|\mathbf{w}\|^2 \tag{B.20}$$

subject to

$$\begin{cases} y_i - \langle \mathbf{w}\cdot\phi(x_i)\rangle - b \le (1-\beta)\epsilon + \xi_i; \\ \langle \mathbf{w}\cdot\phi(x_i)\rangle + b - y_i \le (1-\beta)\epsilon + \xi_i^*; \\ \xi_i \ge 0; \quad \xi_i^* \ge 0 \quad \forall i \end{cases} \tag{B.21}$$

where

$$\psi_s(\xi) = \begin{cases} \dfrac{\xi^2}{4\beta\epsilon} & if \quad \xi \in [0, 2\beta\epsilon]; \\ \xi - \beta\epsilon & if \quad \xi \in (2\beta\epsilon, +\infty). \end{cases} \tag{B.22}$$

Let $\alpha_i \ge 0$, $\alpha_i^* \ge 0$, $\gamma_i \ge 0$ and $\gamma_i^* \ge 0$ $\forall i$ be the corresponding Lagrangian multipliers for the inequalities in (B.21). The KKT conditions for the *primal* problem require

$$\mathbf{w} = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\cdot\phi(x_i) \tag{B.23}$$

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \tag{B.24}$$

$$C\cdot\frac{\partial\psi_e(\xi_i)}{\partial\xi_i} = \alpha_i + \gamma_i \quad \forall i \tag{B.25}$$

$$C\cdot\frac{\partial\psi_e(\xi_i^*)}{\partial\xi_i^*} = \alpha_i^* + \gamma_i^* \quad \forall i \tag{B.26}$$

Based on the definition of ψ_s given by (B.22), the constraint condition (B.26) could be explicitly written as equality constraints:

$$\begin{aligned} &\alpha_i + \gamma_i = C\cdot\frac{\xi_i}{2\beta\epsilon} \quad if \quad 0 \le \xi_i \le 2\beta\epsilon \\ &\alpha_i + \gamma_i = C \qquad\quad if \quad \xi_i > 2\beta\epsilon \end{aligned} \tag{B.27}$$

$$\begin{aligned} &\alpha_i^* + \gamma_i^* = C\cdot\frac{\xi_i^*}{2\beta\epsilon} \quad if \quad 0 \le \xi_i^* \le 2\beta\epsilon \\ &\alpha_i^* + \gamma_i^* = C \qquad\quad if \quad \xi_i^* > 2\beta\epsilon \end{aligned} \tag{B.28}$$

Following the analogous arguments as SVC did in (B.13), we can also eliminate ξ_i and ξ_i^* here.

That yields a maximization problem in terms of the positive *dual* variables $\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*$:

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

subject to $\alpha_i \geq 0$, $\alpha_i^* \geq 0$, $\gamma_i \geq 0$, $\gamma_i^* \geq 0$, $0 \leq \alpha_i + \gamma_i \leq C, \forall i$, $0 \leq \alpha_i^* + \gamma_i^* \leq C, \forall i$ and $\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$. As γ_i and γ_i^* only appear in the last term in (B.29), the functional (B.29) is maximal when $\gamma_i = 0$ and $\gamma_i^* = 0\ \forall i$. Thus, the *dual* problem can be simplified as

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

subject to $0 \leq \alpha_i \leq C, \forall i$, $0 \leq \alpha_i^* \leq C, \forall i$ and $\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$. With the equality (B.23), the dual form of the regression function can be written as $f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x_i, x_j) + b$.

Like SILF, the *dual* problem (B.30) is a generalization of the several SVR formulations. More exactly,

1. when we set $\beta = 0$, (B.30) becomes the classical SVR using ϵ-insensitive loss function;
2. When $\beta = 1$, (B.30) becomes that when the Huber's loss function is used.
3. When $\beta = 0$ and $\epsilon = 0$, (B.30) becomes that for the case of the Laplacian loss function.
4. As for the optimization problem (B.1) using Gaussian loss function with the variance σ^2, that is equivalent to the general SVR (B.30) with $\beta = 1$ and $\frac{2\epsilon}{C} = \sigma^2$ 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.

The *dual* problem (B.30) is also a constrained convex quadratic programming problem. Let us denote

$$\begin{aligned}\hat{\boldsymbol{\alpha}} &= [\alpha_1, \ldots, \alpha_n, -\alpha_1^*, \ldots, -\alpha_n^*]^T, \\ \boldsymbol{P} &= [-y_1 + (1-\beta)\epsilon, \ldots, -y_n + (1-\beta)\epsilon, -y_1 - (1-\beta)\epsilon, \ldots, -y_n - (1-\beta)\epsilon]^T \\ \boldsymbol{Q} &= \begin{bmatrix} \boldsymbol{K} + \frac{2\beta\epsilon}{C}\boldsymbol{I} & \boldsymbol{K} \\ \boldsymbol{K} & \boldsymbol{K} + \frac{2\beta\epsilon}{C}\boldsymbol{I} \end{bmatrix}\end{aligned}$$

where $\boldsymbol{K}$ is the kernel matrix with ij-entry $K(x_i, x_j)$, (B.30) can be rewritten as

$$\min_{\hat{\boldsymbol{\alpha}}} \frac{1}{2}\hat{\boldsymbol{\alpha}}^T \boldsymbol{Q}\hat{\boldsymbol{\alpha}} + \boldsymbol{P}^T\hat{\boldsymbol{\alpha}} \tag{B.31}$$

subject to $l_i \leq \hat{\alpha}_i \leq u_i$, $\forall i$ and $\sum_{i=1}^{2n} \hat{\alpha}_i = 0$ where $l_i = 0$, $u_i = C$ for $1 \leq i \leq n$ and $l_i = -C$, $u_i = 0$ for $n+1 \leq i \leq 2n$. As to the algorithm design for the solution, popular SMO algorithms could be easily adapted for the solution (Smola and Schölkopf, 1998; Shevade et al., 2000) . For a program design and source code, refer to Chu et al. (2001a).

Appendix C

Sequential Minimal Optimization and its Implementation

Sequential minimal optimization (SMO) was proposed by Platt (1999) for support vector classifier. The idea to fix the size of working set at two is an extreme case of the "chunking" idea. The merit of SMO lies in that the sub-optimization problem can be solved analytically. Keerthi et al. (2001) put forwards improvements on SMO by introducing two thresholds to determine the bias. In this chapter, we adapt the popular SMO algorithm to solve the constrained quadratic programming (QP) problems arising in the general formulation for SVM.

C.1 Optimality Conditions

We briefly study the optimality conditions for the solution of the QP problems, (B.18) or (B.31). By introducing Lagrange multipliers for the constraints in the QP, we have to find the saddle of the Lagrange functional:

$$\mathbf{L} = \frac{1}{2}\hat{\boldsymbol{\alpha}}^T \boldsymbol{Q}\hat{\boldsymbol{\alpha}} + \boldsymbol{P}^T\hat{\boldsymbol{\alpha}} - \sum_i \eta_i(\hat{\alpha}_i - l_i) + \sum_i \lambda_i(\hat{\alpha}_i - u_i) + \gamma \sum_i \hat{\alpha}_i \tag{C.1}$$

Let us define

$$F_i(\hat{\boldsymbol{\alpha}}) = -[\boldsymbol{Q}\hat{\boldsymbol{\alpha}}]_i - \boldsymbol{P}_i \tag{C.2}$$

where $[\boldsymbol{Q}\hat{\boldsymbol{\alpha}}]_i$ denotes the i-th entry of the vector $\boldsymbol{Q}\hat{\boldsymbol{\alpha}}$ and $\boldsymbol{P}_i$ is the i-th entry of the vector $\boldsymbol{P}$. The KKT conditions for the QP are:

$$\begin{aligned} &\tfrac{\partial \mathbf{L}}{\partial \hat{\boldsymbol{\alpha}}_i} = -F_i(\hat{\boldsymbol{\alpha}}) - \eta_i + \lambda_i + \gamma \\ &\eta_i \geq 0, \eta_i(\hat{\boldsymbol{\alpha}}_i - l_i) = 0, \lambda_i \geq 0, \lambda_i(\hat{\boldsymbol{\alpha}}_i - u_i) = 0, \forall i \end{aligned} \tag{C.3}$$

These conditions can be simplified by considering three cases for each i:

$$\begin{aligned} &\text{Case 1}: \quad \hat{\boldsymbol{\alpha}}_i = l_i \quad F_i(\hat{\boldsymbol{\alpha}}) - \gamma \leq 0 \\ &\text{Case 2}: \quad l_i < \hat{\boldsymbol{\alpha}}_i < u_i \quad F_i(\hat{\boldsymbol{\alpha}}) - \gamma = 0 \\ &\text{Case 3}: \quad \hat{\boldsymbol{\alpha}}_i = u_i \quad F_i(\hat{\boldsymbol{\alpha}}) - \gamma \geq 0 \end{aligned} \tag{C.4}$$

Let us define three index sets: $I_0(\hat{\boldsymbol{\alpha}}) = \{i : l_i < \hat{\boldsymbol{\alpha}}_i < u_i\}$; $I_1(\hat{\boldsymbol{\alpha}}) = \{i : \hat{\boldsymbol{\alpha}}_i = u_i\}$ and $I_2(\hat{\boldsymbol{\alpha}}) = \{i : \hat{\boldsymbol{\alpha}}_i = l_i\}$, and then further $I_{up}(\hat{\boldsymbol{\alpha}}) = I_0(\hat{\boldsymbol{\alpha}}) \cup I_1(\hat{\boldsymbol{\alpha}})$ and $I_{low}(\hat{\boldsymbol{\alpha}}) = I_0(\hat{\boldsymbol{\alpha}}) \cup I_2(\hat{\boldsymbol{\alpha}})$. The optimality conditions can be written as

$$\begin{aligned} &\gamma \leq F_i(\hat{\boldsymbol{\alpha}}) \quad \forall\, i \in I_{up}(\hat{\boldsymbol{\alpha}}) \\ &\gamma \geq F_i(\hat{\boldsymbol{\alpha}}) \quad \forall\, i \in I_{low}(\hat{\boldsymbol{\alpha}}) \end{aligned} \tag{C.5}$$

We introduce a positive tolerance parameter τ to define approximate optimality conditions. We will say that (i, j) is a τ-violating pair at $\hat{\boldsymbol{\alpha}}$ if one of the following two sets of events occurs:

$$\begin{aligned} &i \in I_{up}(\hat{\boldsymbol{\alpha}}), \quad j \in I_{low}(\hat{\boldsymbol{\alpha}}) \quad \text{and} \quad F_i(\hat{\boldsymbol{\alpha}}) < F_j(\hat{\boldsymbol{\alpha}}) - \tau \\ &i \in I_{low}(\hat{\boldsymbol{\alpha}}), \quad j \in I_{up}(\hat{\boldsymbol{\alpha}}) \quad \text{and} \quad F_i(\hat{\boldsymbol{\alpha}}) > F_j(\hat{\boldsymbol{\alpha}}) + \tau \end{aligned} \tag{C.6}$$

Define $b_{up}(\hat{\boldsymbol{\alpha}}) = \min\{F_i(\hat{\boldsymbol{\alpha}}) : i \in I_{up}(\hat{\boldsymbol{\alpha}})\}$ and $b_{low}(\hat{\boldsymbol{\alpha}}) = \max\{F_i(\hat{\boldsymbol{\alpha}}) : i \in I_{low}(\hat{\boldsymbol{\alpha}})\}$. The optimality conditions can be compactly written as

$$b_{up} \geq b_{low} - \tau \tag{C.7}$$

If (C.7) holds, we say that $\hat{\boldsymbol{\alpha}}$ is a τ-optimal solution, and then the bias b in *primal* problem can be determined as $\frac{b_{up}+b_{low}}{2}$. Note that the training samples whose index $i \in I_0$ is just the set of off-bound SVs.

So far, we have got two conditions in (C.6) for checking optimality, and a stopping condition (C.7). Following the design in Keerthi et al. (2001), we employ a two-loop approach till the stopping condition is satisfied. The *Type I* loop sweeps over all the samples one by one to check

optimality with current b_{up} or b_{low},[1] and then updates the pair of samples who are violating the optimality conditions (C.6). While the *Type II* loop update the pair associated with b_{up} and b_{low}, and then vote new b_{up} and b_{low} within the I_0 set. The *Type II* loop are repeated till (C.7) are satisfied or no sample can be updated, then the *Type I* loop is executed to check the optimality violation for all samples again. We describe the SMO algorithm in Table C.1:

Table C.1: The main loop in SMO algorithm.

SMO	**Algorithm**
Initialization	BOOL CheckAll = TRUE ;
	BOOL Violation = TRUE ;
	choose any $\hat{\boldsymbol{\alpha}}$ that satisfies $l_i \leq \hat{\boldsymbol{\alpha}}_i \leq u_i$
	vote the current b_{up} and b_{low}
	While (CheckAll == TRUE or Violation == TRUE)
	if (CheckAll == TRUE)
Type I Loop	check optimality condition one by one over all samples
	update the violating pair and then set Violation = TRUE
	else
Type II Loop	while (the stopping condition does not hold)
	update the pair associated with b_{up} and b_{low};
	vote b_{up} and b_{low} within the set I_0
	endwhile
	set Violation = FALSE
	endif
Switcher	if (CheckAll == TRUE)
	set CheckAll = FALSE
	elseif (Violation == FALSE)
	set CheckAll = TRUE
	endif
	endwhile
Termination	return $\hat{\boldsymbol{\alpha}}$

C.2 Sub-optimization Problem

Now we study the solution to the sub-optimization problem, i.e. how to update the violating pair. We update two Lagrange multipliers towards the optimal values in either *Type I* or *Type II* loop every time. Suppose that the pair of the Lagrangian multipliers being updated are $\hat{\boldsymbol{\alpha}}_i$ and $\hat{\boldsymbol{\alpha}}_j$. The other Lagrangian multipliers are fixed during the updating. Thus, we only need to find the minimization solution to a sub-optimization problem. The sub-optimization problem for the QP problem can be given as

$$\min_{\hat{\boldsymbol{\alpha}}_i, \hat{\boldsymbol{\alpha}}_j} S(\hat{\boldsymbol{\alpha}}_i, \hat{\boldsymbol{\alpha}}_j) = \frac{1}{2}\hat{\boldsymbol{\alpha}}^T \boldsymbol{Q}\hat{\boldsymbol{\alpha}} + \boldsymbol{P}_i\hat{\boldsymbol{\alpha}}_i + \boldsymbol{P}_j\hat{\boldsymbol{\alpha}}_j \tag{C.8}$$

[1] If the sample being checked belongs to $I_{up}(\hat{\boldsymbol{\alpha}})$, we check it with b_{low}; otherwise, we check it with b_{up}.

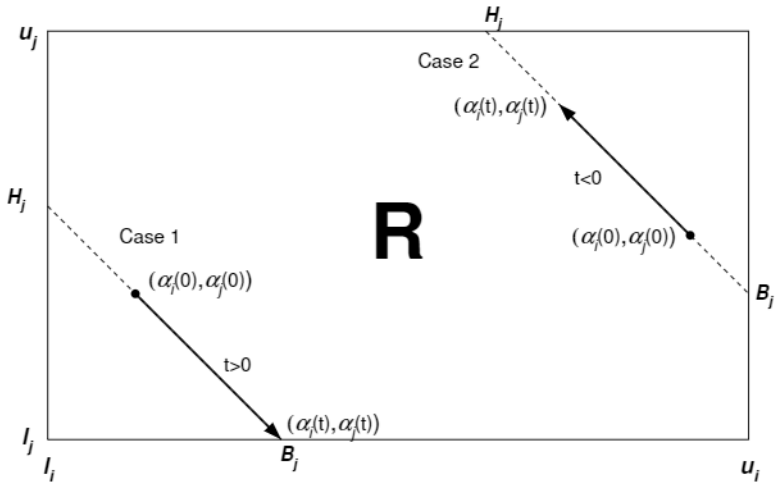

Figure C.1: Minimization steps within the rectangle $R = [l_i, u_i] \times [l_j, u_j]$ in the sub-optimization problem.

subject to $l_i \leq \hat{\boldsymbol{\alpha}}_i \leq u_i$, $l_j \leq \hat{\boldsymbol{\alpha}}_j \leq u_j$ and $\hat{\boldsymbol{\alpha}}_i + \hat{\boldsymbol{\alpha}}_j = c$ where c is a constant. Clearly, the minimization of the sub-optimization problem (C.8) takes place in the rectangle $R = [l_i, u_i] \times [l_j, u_j]$ along the path determined by $\hat{\boldsymbol{\alpha}}_i + \hat{\boldsymbol{\alpha}}_j = c$, which is a straight line with negative unit slope, as shown in Figure C.1. Let us denote the current Lagrange multipliers as $\hat{\boldsymbol{\alpha}}_i(0)$ and $\hat{\boldsymbol{\alpha}}_j(0)$, and then define the potential solution to (C.8) as $\hat{\boldsymbol{\alpha}}_i(t)$ and $\hat{\boldsymbol{\alpha}}_j(t)$, where $\hat{\boldsymbol{\alpha}}_i(t) = \hat{\boldsymbol{\alpha}}_i(0) + t$ and $\hat{\boldsymbol{\alpha}}_j(t) = \hat{\boldsymbol{\alpha}}_j(0) - t$ and $t \in \mathbb{R}$. Our objective is to minimize the functional $S(t) = S(\hat{\boldsymbol{\alpha}}_i(t), \hat{\boldsymbol{\alpha}}_j(t))$ subject to the pair $(\hat{\boldsymbol{\alpha}}_i, \hat{\boldsymbol{\alpha}}_j) \in R$. It is easy to confirm that $S(t) = S(0) + S'(0)t + \frac{1}{2}S''(0)t^2$ where $S(0) = S(\hat{\boldsymbol{\alpha}}_i(0), \hat{\boldsymbol{\alpha}}_j(0))$, $S'(0) = F_j - F_i$ and $S''(0) = \boldsymbol{Q}_{ii} - 2\boldsymbol{Q}_{ij} + \boldsymbol{Q}_{jj}$.[2] By the semi-positive definiteness of $\boldsymbol{Q}$ it follows $S''(0) \geq 0$. The unconstrained minimum of $S(t)$ should be at $t_u = -\frac{S'(0)}{S''(0)}$, i.e. the unconstrained solution should be $\hat{\boldsymbol{\alpha}}_i(t) = \hat{\boldsymbol{\alpha}}_i(0) + t_u$ and $\hat{\boldsymbol{\alpha}}_j(t) = \hat{\boldsymbol{\alpha}}_j(0) - t_u$. We now take the rectangle constraint into account by holding $B_j \leq \hat{\boldsymbol{\alpha}}_j(t) \leq H_j$, where $B_j = \max\{c - u_i, l_j\}$ and $H_j = \min\{c - l_i, u_j\}$.

In the implementation, we cache F_i for each sample who lies in the set I_0. [3] This helps especially when the *type II loop* is executing, as F_i are used quite frequently. After solving the sub-optimization problem, we update F_i in the cache by

$$F_k^{new} = F_k^{old} + \big(\hat{\boldsymbol{\alpha}}_i(0) - \hat{\boldsymbol{\alpha}}_i(t)\big)\boldsymbol{Q}_{ik} + \big(\hat{\boldsymbol{\alpha}}_j(0) - \hat{\boldsymbol{\alpha}}_j(t)\big)\boldsymbol{Q}_{jk} \qquad \forall k \tag{C.9}$$

[2] F_i has been defined as in (C.2) at $\hat{\boldsymbol{\alpha}}(0)$.
[3] We also cache the diagonal entries of $\boldsymbol{Q}$, i.e. $\boldsymbol{Q}_{ii} \forall i$.

C.3 Conjugate Enhancement in Regression

Although the SMO algorithm we described in last section could be used to solve the QP problem in SVR (B.31), it is not an efficient way since we have to double the QP size. Notice that the constraints $\alpha_i\alpha_i^* = 0\ \forall i$ always exist since both of them are associated with the sample (x_i, y_i). In the SMO algorithm for standard SVR (Smola and Schölkopf, 1998; Shevade et al., 2000), the constraints $\alpha_i\alpha_i^* = 0$ are taken into account in minimization steps. Pairs of variables (α_i, α_i^*) are selected simultaneously into the working set, and then consider the possible results in four quadrants.

Optimality Conditions The Lagrangian for the *dual* problem (B.30) is defined as:

$$\begin{aligned}\mathbf{L} = \ & \tfrac{1}{2}\textstyle\sum_{i=1}^n\sum_{j=1}^n(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\boldsymbol{Q}_{ij} - \sum_{i=1}^n y_i(\alpha_i-\alpha_i^*) \\ & + \textstyle\sum_{i=1}^n(1-\beta)\epsilon(\alpha_i+\alpha_i^*) - \sum_{i=1}^n \pi_i\alpha_i - \sum_{i=1}^n \psi_i\alpha_i^* \\ & - \textstyle\sum_{i=1}^n \lambda_i(C-\alpha_i) - \sum_{i=1}^n \eta_i(C-\alpha_i^*) + \gamma\sum_{i=1}^n(\alpha_i-\alpha_i^*)\end{aligned} \tag{C.10}$$

where $\boldsymbol{Q}_{ij} = \langle\phi(x_i)\cdot\phi(x_j)\rangle + \delta_{ij}\frac{2\beta\epsilon}{C}$ with δ_{ij} the Kronecker delta. Let us define

$$F_i = y_i - \sum_{j=1}^n(\alpha_i-\alpha_i^*)Q(x_i,x_j) \tag{C.11}$$

and the KKT conditions should be:

$$\begin{aligned}&\tfrac{\partial\mathbf{L}}{\partial\alpha_i} = -F_i + (1-\beta)\epsilon - \pi_i + \lambda_i + \gamma = 0 \\ &\pi_i \geq 0, \pi_i\alpha_i = 0, \lambda_i \geq 0, \lambda_i(C-\alpha_i) = 0, \forall i \\ &\tfrac{\partial\mathbf{L}}{\partial\alpha_i^*} = F_i + (1-\beta)\epsilon - \psi_i + \eta_i - \gamma = 0 \\ &\psi_i \geq 0, \psi_i\alpha_i^* = 0, \eta_i \geq 0, \eta_i(C-\alpha_i^*) = 0, \forall i\end{aligned} \tag{C.12}$$

These conditions can be simplified by considering five cases for each i:

$$\begin{aligned}&\text{Case 1}: \quad \alpha_i = \alpha_i^* = 0 \quad -(1-\beta)\epsilon \leq F_i - \gamma \leq (1-\beta)\epsilon \\ &\text{Case 2}: \quad \alpha_i = C \qquad\qquad F_i - \gamma \geq (1-\beta)\epsilon \\ &\text{Case 3}: \quad \alpha_i^* = C \qquad\qquad F_i - \gamma \leq -(1-\beta)\epsilon \\ &\text{Case 4}: \quad 0 < \alpha_i < C \qquad\qquad F_i - \gamma = (1-\beta)\epsilon \\ &\text{Case 5}: \quad 0 < \alpha_i^* < C \qquad\qquad F_i - \gamma = -(1-\beta)\epsilon\end{aligned} \tag{C.13}$$

We can classify any one pair into one of the following five sets, which are defined as:

$$\begin{aligned} I_{0a} &= \{i : 0 < \alpha_i < C\} \\ I_{0b} &= \{i : 0 < \alpha_i^* < C\} \\ I_0 &= I_{0a} \cup I_{0b} \\ I_1 &= \{i : \alpha_i = \alpha_i^* = 0\} \\ I_2 &= \{i : \alpha_i^* = C\} \\ I_3 &= \{i : \alpha_i = C\} \end{aligned} \tag{C.14}$$

Let us denote $I_{up} = I_0 \cup I_1 \cup I_3$ and $I_{low} = I_0 \cup I_1 \cup I_2$. We further define F_i^{up} on the set I_{up} as

$$F_i^{up} = \begin{cases} F_i + (1-\beta)\epsilon & \text{if} \quad i \in I_{0b} \cup I_1 \\ F_i - (1-\beta)\epsilon & \text{if} \quad i \in I_{0a} \cup I_3 \end{cases} \tag{C.15}$$

and F_i^{low} on the set I_{low} as

$$F_i^{low} = \begin{cases} F_i + (1-\beta)\epsilon & \text{if} \quad i \in I_{0b} \cup I_2 \\ F_i - (1-\beta)\epsilon & \text{if} \quad i \in I_{0a} \cup I_1 \end{cases} \tag{C.16}$$

Then the conditions in (C.13) can be simplified as

$$\gamma \leq F_i^{up} \ \forall i \in I_{up} \quad \text{and} \quad \gamma \geq F_i^{low} \ \forall i \in I_{low} \tag{C.17}$$

Thus the stopping condition can be compactly written as:

$$b_{up} \geq b_{low} - \tau \tag{C.18}$$

where $b_{up} = \min\{F_i^{up} : i \in I_{up}\}$, $b_{low} = \max\{F_i^{low} : i \in I_{low}\}$, and the tolerance parameter $\tau > 0$. If (C.18) holds, we reach a τ-optimal solution, and then the bias b can also be determined as $\frac{b_{up}+b_{low}}{2}$. At the optimal solution, the training samples whose index $i \in I_0$ are usually called off-bound SVs, on-bound SVs if $i \in I_2 \cup I_3$, and non-SVs if $i \in I_1$.

Sub-optimization Problem Suppose that the i-th and j-th samples are collected in the working set, and denote $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^T$ and $\boldsymbol{\alpha}^* = [\alpha_1^*, \ldots, \alpha_n^*]^T$. The sub-optimization

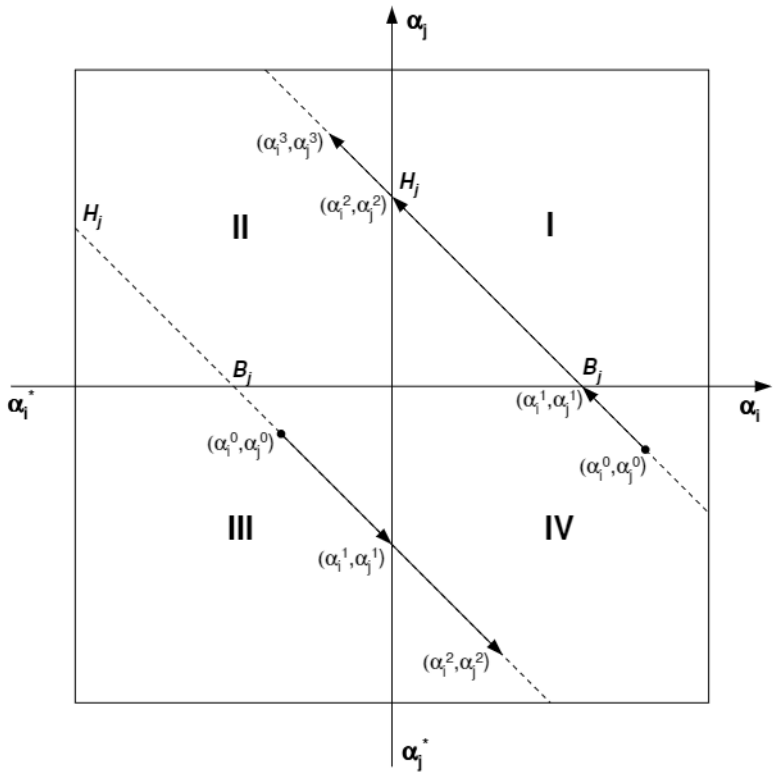

Figure C.2: Possible quadrant changes of the pair of the Lagrange multipliers.

problem can be written as

$$\begin{aligned}\min_{\alpha_i,\alpha_i^*,\alpha_j,\alpha_j^*} \quad & S(\alpha_i,\alpha_i^*,\alpha_j,\alpha_j^*) = \tfrac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T\boldsymbol{Q}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & -(\alpha_i-\alpha_i^*)y_i - (\alpha_j-\alpha_j^*)y_j + (1-\beta)\epsilon(\alpha_i+\alpha_i^*+\alpha_j+\alpha_j^*)\end{aligned} \tag{C.19}$$

subject to $0 \le \alpha_i \le C$, $0 \le \alpha_i^* \le C$, $0 \le \alpha_j \le C$, $0 \le \alpha_j^* \le C$ and $\alpha_i - \alpha_i^* + \alpha_j - \alpha_j^* = \nu$ where ν is a constant. We have to distinguish four different cases: (α_i, α_j), (α_i, α_j^*), (α_i^*, α_j), (α_i^*, α_j^*). It is easy to derive the unconstrained solution to the sub-optimization problem at the four quadrants and the boundary of feasible regions. We tabulate these results in Table C.2, where $\rho = \boldsymbol{Q}_{ii} - 2\boldsymbol{Q}_{ij} + \boldsymbol{Q}_{jj}$ and the linear constraint $\alpha_i - \alpha_i^* + \alpha_j - \alpha_j^* = \nu$ should hold.

Table C.2: Boundary of feasible regions and unconstrained solution in the four quadrants, where $\rho = \boldsymbol{Q}_{ii} - 2\boldsymbol{Q}_{ij} + \boldsymbol{Q}_{jj}$.

Quadrant	Boundary of α_j or α_j^*	Unconstrained Solution
I (α_i, α_j)	$H_j = \min(C, \nu)$; $B_j = \max(0, \nu - C)$	$\alpha_j^{new} = \alpha_j + (F_j - F_i)/\rho$
II (α_i^*, α_j)	$H_j = \min(C, C + \nu)$; $B_j = \max(0, \nu)$	$\alpha_j^{new} = \alpha_j + (F_j - F_i - 2(1-\beta)\epsilon)/\rho$
III (α_i^*, α_j^*)	$H_j = \min(C, -\nu)$; $B_j = \max(0, -\nu - C)$	$\alpha_j^{*new} = \alpha_j + (F_i - F_j)/\rho$
IV (α_i, α_j^*)	$H_j = \min(C - \nu, C)$; $B_j = \max(0, -\nu)$	$\alpha_j^{*new} = \alpha_j + (F_i - F_j - 2(1-\beta)\epsilon)/\rho$

It may happen that for a fixed pair of indices (i, j) the initial chosen quadrant, say e.g. (α_i, α_j^*) is the one with optimal solution. In a particular case the other quadrants, (α_i, α_j) and even (α_i^*, α_j), have to be checked. It occurs if one of the two variables hits the 0 boundary, and then the computation of the corresponding values for the variables with(out) asterisk according to the following table is required. The possible quadrant transfers are shown in Figure C.2. Thus we have to compute $F_i^{new} - F_j^{new}$ after an updating, which is equal to

$$F_i^{new} - F_j^{new} = F_i^{old} - F_j^{old} - \rho(\alpha_i^{new} - \alpha_i^{*new} - \alpha_i^{old} + \alpha_i^{*old}) \tag{C.20}$$

Sometimes it may happen that $\rho \leq 0$. In that case, the optimal values lie on the boundaries H_j or B_j. We can determine simply which endpoints should be taken by computing the value of the objective function at these endpoints.

C.4 Implementation in ANSI C

In this section, we report a program design to implement the SMO algorithm in ANSI C. We describe the data structure design first that is the basis even for other learning algorithm, and then introduce the main functions of other routines.

Design on Data Structure Data structures are the basis for an algorithm implementation. We design three key data structures to save and manipulate the training data.

1. **Pairs**, which is a list to save training (or test) samples. Each node contains the input vector and the target value of a training sample.

2. **Alpha**, which is a $n \times 1$ structure vector. Each structure contains a pointer reference to the associated pair node (x_i, y_i), the current value of α_i (and α_i^* for regression) and the associated upper/lower bounds, the current set name which is determined by α_i value, the current value of F_i if the set name is just I_0 and a pointer reference to the corresponding cache node.

3. I_0**_Cache**, which is a list of F_i for the set I_0, i.e. the current set of off-bound SVs. Each node contains a reference of the pointer of the **Alpha** structure in which we can access the current F_i.

4. **smo_Settings**, which is a structure to save the settings of SVM that contains all necessary parameters that control the behavior of the SMO algorithm.

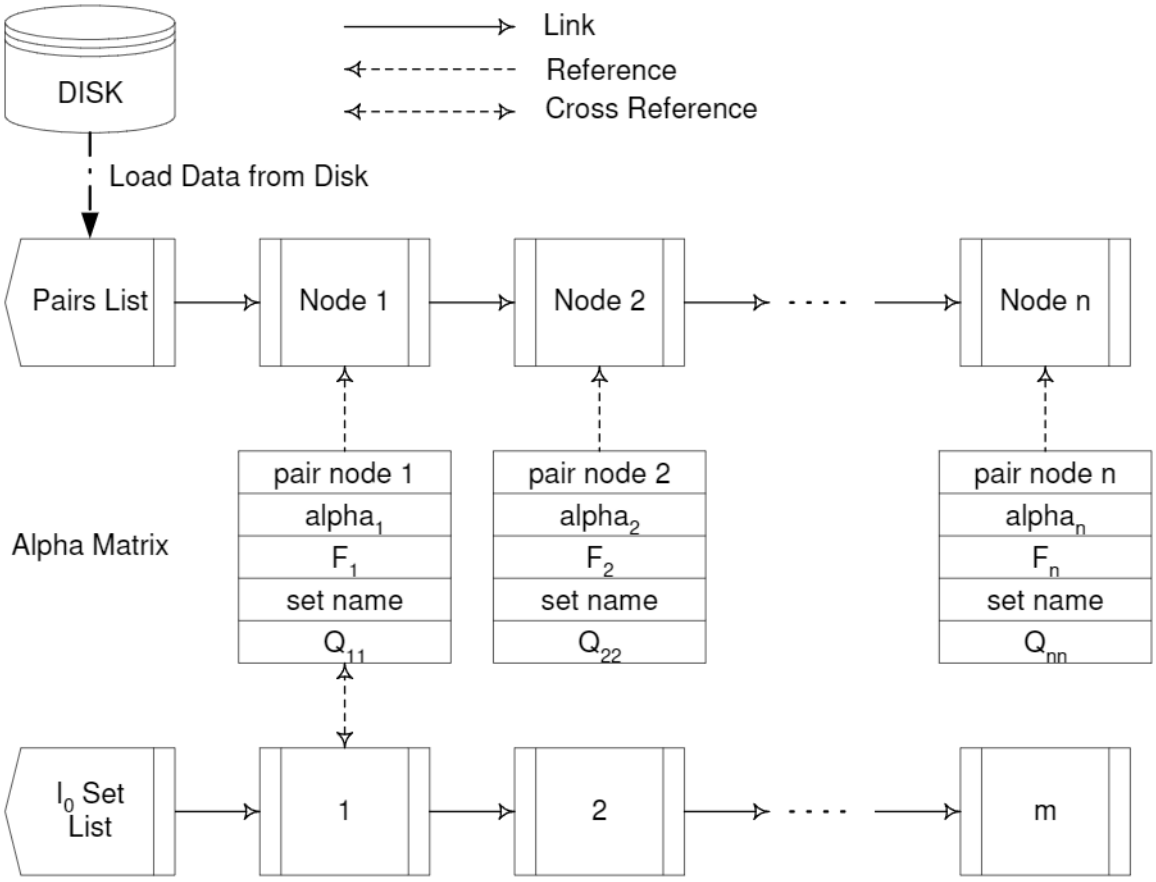

Figure C.3: Relationship of Data Structures.

Their inter-relationship is presented in Figure C.3. The manipulations on these data structures have been implemented in ANSI C, whose source code can be found in datalist.cpp, alphas.cpp, cachelist.cpp and smo_settings.cpp respectively. The declarations are collected in smo.h.[4]

Routine Description There are several routines, each of which serves a specified function, to cooperate to implement the algorithm. The program we implemented is strictly compatible with the data format of DELVE.[5]

1. BOOL **smo_Loadfile**, which loads data file from disk to initialize the **Pairs** list. The full name of the data file has already been saved in the structure **smo_Settings** after creating the structure.

2. BOOL **smo_Routine**, which is the main loop of the SMO algorithm.

[4]The source code can be accessed at http://guppy.mpe.nus.edu.sg/~chuwei/smo/usmo.zip.
[5]DELVE, data for evaluating learning in valid experiments, contains a lot of benchmark data sets, available at http://www.cs.toronto.edu/~delve/.

3. BOOL **smo_ExamineSample**, which checks the optimality condition of the sample with current b_{up} or b_{low}.

4. BOOL **smo_TakeStep**, which updates the sample pair by analytically solving the sub-optimization problem.

5. BOOL **smo_Prediction**, which calculates the test error if the test data are available.

6. BOOL **smo_Dumping**, which creates log files to save the result.

7. BOOL **Calc_Kernel**, which calculate the kernel value of two input vectors.

Appendix D

Proof of Parameterization Lemma

The following property of Gaussian distribution will be used to prove the Parameterization Lemma, we first state it separately:

Property of Gaussian Distribution (Csató and Opper, 2002). Let $\boldsymbol{f} \in \mathbb{R}^n$ and a general Gaussian probability density function (pdf)

$$\mathcal{P}(\boldsymbol{f}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{f}-\boldsymbol{f}_0)^T\Sigma^{-1}(\boldsymbol{f}-\boldsymbol{f}_0)\right)$$

where the mean $\boldsymbol{f}_0 \in \mathbb{R}^n$ and the $n \times n$ covariance matrix Σ with ij-th entry $Cov[f_i, f_j]$ $\forall i, j$. If $g : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function not growing faster than a polynomial and with partial derivatives $\nabla g(\boldsymbol{f}) = \dfrac{\partial}{\partial \boldsymbol{f}} g(\boldsymbol{f})$, and then[1]

$$\int \boldsymbol{f}\, \mathcal{P}(\boldsymbol{f}) g(\boldsymbol{f}) d\boldsymbol{f} = \boldsymbol{f}_0 \int \mathcal{P}(\boldsymbol{f})\, g(\boldsymbol{f}) d\boldsymbol{f} + \Sigma \int \mathcal{P}(\boldsymbol{f})\, \nabla g(\boldsymbol{f}) d\boldsymbol{f} \tag{D.1}$$

Proof. The proof uses the partial integration rule:

$$\int \mathcal{P}(\boldsymbol{f}) \nabla g(\boldsymbol{f}) d\boldsymbol{f} = -\int g(\boldsymbol{f}) \nabla \mathcal{P}(\boldsymbol{f}) d\boldsymbol{f}.$$

Using the derivative of a Gaussian pdf we have:

$$\int \mathcal{P}(\boldsymbol{f}) \nabla g(\boldsymbol{f}) d\boldsymbol{f} = \int \mathcal{P}(\boldsymbol{f}) g(\boldsymbol{f}) \Sigma^{-1}(\boldsymbol{f}-\boldsymbol{f}_0) d\boldsymbol{f} \tag{D.2}$$

Multiplying both sides with Σ leads to (D.1). □

[1] In the following we will assume definite integral over the $\boldsymbol{f}$-space whenever the integral appears.

In Parameterization Lemma, we are interested in computing the mean function of the posterior distribution in the Gaussian process:

$$\begin{aligned} E[f(x)]_n &= \int f(x)\mathcal{P}(\bar{\boldsymbol{f}}|\mathcal{D})d\bar{\boldsymbol{f}} = \frac{1}{\mathcal{P}(\mathcal{D})}\int f(x)\mathcal{P}(\mathcal{D}|\bar{\boldsymbol{f}})\mathcal{P}(\bar{\boldsymbol{f}})d\bar{\boldsymbol{f}} \\ &= \frac{1}{\mathcal{P}(\mathcal{D})}\int f(x)\mathcal{P}(\bar{\boldsymbol{f}})\mathcal{P}(\mathcal{D}|\boldsymbol{f})d\bar{\boldsymbol{f}} \end{aligned}$$

where $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, $\bar{\boldsymbol{f}} = [f(x), \boldsymbol{f}]^T$ and $\boldsymbol{f} = [f(x_1), \ldots, f(x_n)]^T$. Let us apply (D.1) replacing $g(x)$ by $\mathcal{P}(\mathcal{D}|\boldsymbol{f})$ and only pick up the entry of $f(x)$ in the vector form, we have

$$E[f(x)]_n = \frac{1}{\mathcal{P}(\mathcal{D})}\left(f_0(x)\int \mathcal{P}(\bar{\boldsymbol{f}})\mathcal{P}(\mathcal{D}|\boldsymbol{f})d\bar{\boldsymbol{f}} + \sum_{i=1}^n Cov[f(x), f(x_i)]\int \mathcal{P}(\bar{\boldsymbol{f}})\frac{\partial \mathcal{P}(\mathcal{D}|\boldsymbol{f})}{\partial f(x_i)}d\bar{\boldsymbol{f}}\right) \tag{D.3}$$

The variable $f(x)$ in the integrals vanishes since it is only contained in $\mathcal{P}(\bar{\boldsymbol{f}})$, (D.3) can be further simplified as

$$E[f(x)]_n = f_0(x) + \sum_{i=1}^n Cov[f(x), f(x_i)] \cdot w(i) \tag{D.4}$$

where

$$w(i) = \frac{1}{\mathcal{P}(\mathcal{D})}\int \mathcal{P}(\boldsymbol{f})\frac{\partial \mathcal{P}(\mathcal{D}|\boldsymbol{f})}{\partial f(x_i)}d\boldsymbol{f} \tag{D.5}$$

Notice that the coefficients $w(i)$ depends only on the training data $\mathcal{D}$ and are independent from x at which the posterior mean is evaluated.

Now we change the variables in the integral of (D.5): $f(x_i)' = f(x_i) - f_0(x_i)$ where $f_0(x_i)$ is the prior mean at x_i and keep other variables unchanged $f(x_j)' = f(x_j), j \neq i$, that leads to the following integral:

$$\int \mathcal{P}(\boldsymbol{f}')\frac{\partial \mathcal{P}(\mathcal{D}|\boldsymbol{f}')}{\partial f(x_i)'}d\boldsymbol{f}' \tag{D.6}$$

where $\boldsymbol{f}' = [f(x_1)', \ldots, f(x_i)' + f_0(x_i), \ldots, f(x_n)']^T$. We change the partial differentiation is with respect to $f(x_i)'$ with the partial differentiation with respect to $f_0(x_i)$ and exchange the differentiation and the integral operators, that leads to

$$\frac{\partial}{\partial f_0(x_i)}\int \mathcal{P}(\boldsymbol{f}')\mathcal{P}(\mathcal{D}|\boldsymbol{f}')d\boldsymbol{f}' \tag{D.7}$$

We then perform the inverse change of the variables inside the integral and substitute back into the expression for $w(i)$, that leads to

$$w(i) = \frac{\partial}{\partial f_0(x_i)}\ln\int \mathcal{P}(\boldsymbol{f})\mathcal{P}(\mathcal{D}|\boldsymbol{f})d\boldsymbol{f} \tag{D.8}$$

The posterior covariance can be written as

$$Cov(x,x')_n = E[f(x)f(x')]_n - E[f(x)]_n \cdot E[f(x')]_n \tag{D.9}$$

Let us apply (D.1) twice on $E[f(x)f(x')]_n$ which is written as

$$E[f(x)f(x')]_n = \frac{1}{\mathcal{P}(\mathcal{D})} \int f(x)f(x)'\mathcal{P}(\hat{\boldsymbol{f}})\mathcal{P}(\mathcal{D}|\boldsymbol{f})d\hat{\boldsymbol{f}} \tag{D.10}$$

where $\hat{\boldsymbol{f}} = [f(x), f(x)', \boldsymbol{f}]^T$, that leads to

$$\begin{aligned}
E[f(x)f(x')]_n &= \frac{1}{\mathcal{P}(\mathcal{D})}\Bigg[f_0(x)\int f(x)'\mathcal{P}(\hat{\boldsymbol{f}})\mathcal{P}(\mathcal{D}|\boldsymbol{f})d\hat{\boldsymbol{f}} + Cov(x,x')\int \mathcal{P}(\hat{\boldsymbol{f}})\mathcal{P}(\mathcal{D}|\boldsymbol{f})d\hat{\boldsymbol{f}} \\
&\quad + \sum_{i=1}^{n} Cov(x,x_i)\int f(x')\mathcal{P}(\hat{\boldsymbol{f}})\frac{\partial \mathcal{P}(\mathcal{D}|\boldsymbol{f})}{\partial f(x_i)}d\hat{\boldsymbol{f}}\Bigg] \\
&= f_0(x)\cdot E[f(x')]_n + \frac{1}{\mathcal{P}(\mathcal{D})}\sum_{i=1}^{n} Cov(x,x_i)\Bigg[f_0(x')\int \mathcal{P}(\hat{\boldsymbol{f}})\frac{\partial \mathcal{P}(\mathcal{D}|\boldsymbol{f})}{\partial f(x_i)}d\hat{\boldsymbol{f}} \\
&\quad + \sum_{j=1}^{n} Cov(x_j,x')\int \mathcal{P}(\hat{\boldsymbol{f}})\frac{\partial^2 \mathcal{P}(\mathcal{D}|\boldsymbol{f})}{\partial f(x_i)\partial f(x_j)}d\hat{\boldsymbol{f}}\Bigg] + Cov(x,x') \\
&= Cov(x,x') + f_0(x)\cdot E[f(x')]_n + f_0(x')\sum_{i=1}^{n} Cov(x,x_i)\cdot w(i) \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{n} Cov(x,x_i)\cdot Cov(x_j,x')\frac{1}{\mathcal{P}(\mathcal{D})}\int \mathcal{P}(\hat{\boldsymbol{f}})\frac{\partial^2 \mathcal{P}(\mathcal{D}|\boldsymbol{f})}{\partial f(x_i)\partial f(x_j)}d\hat{\boldsymbol{f}}
\end{aligned} \tag{D.11}$$

Since $E[f(x')]_n = f_0(x') + \sum_{j=1}^{n} Cov(x',x_j)\cdot w(j)$ from (D.4) and the variables $f(x)$ and $f(x')$ will disappear in the integral, (D.11) can be finally written as

$$\begin{aligned}
E[f(x)f(x')]_n &= Cov(x,x') + E[f(x)]_n \cdot E[f(x')]_n \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{n} Cov(x,x_i)\cdot Cov(x_j,x')\big[Q(ij) - w(i)w(j)\big]
\end{aligned} \tag{D.12}$$

where $Q(ij) = \dfrac{1}{\mathcal{P}(\mathcal{D})}\displaystyle\int \mathcal{P}(\boldsymbol{f})\frac{\partial^2 \mathcal{P}(\mathcal{D}|\boldsymbol{f})}{\partial f(x_i)\partial f(x_j)}d\boldsymbol{f}$.

Therefore, the posterior covariance can be given as

$$Cov(x,x')_n = Cov(x,x') + \sum_{i=1}^{n}\sum_{j=1}^{n} Cov(x,x_i)\cdot\big[Q(ij) - w(i)w(j)\big]\cdot Cov(x_j,x') \tag{D.13}$$

Note that

$$R(ij) = Q(ij) - w(i)w(j) = \frac{\partial^2}{\partial f(x_i)\partial f(x_j)}\ln\int \mathcal{P}(\mathcal{D}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})d\boldsymbol{f} \tag{D.14}$$

holds. Now we perform the change of variables in the integral for $Q(ij)$, i.e., repeat the steps from (D.5) to (D.8), that finally leads to

$$R(ij) = \frac{\partial^2}{\partial f_0(x_i)\partial f_0(x_j)} \ln \int \mathcal{P}(\mathcal{D}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})d\boldsymbol{f} \tag{D.15}$$

and using a single training sample in the likelihood leads to the scalar coefficients $w(k+1)$ and $r(k+1)$ as in (3.65) for the on-line formulation (3.62) and (3.63).

Appendix E

Some Derivations

In this chapter, we give more details about the derivations for some important equations in previous chapters.

E.1 The Derivation for Equation (4.37)

The exact evaluation on the evidence is an integral over the space of $\boldsymbol{f}$ as given in (4.35). We try to give an explicit expression for evidence evaluation by resorting to Laplacian approximation, i.e. Taylor expansion of $S(\boldsymbol{f})$ around $S(\boldsymbol{f}_{\text{MP}})$ up to the second order:

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

At the MAP estimate point $\boldsymbol{f}_{\text{MP}}$, $\left.\frac{\partial S(\boldsymbol{f})}{\partial \boldsymbol{f}}\right|_{\boldsymbol{f}=\boldsymbol{f}_{\text{MP}}} = 0$ holds, and $\left.\frac{\partial^2 S(\boldsymbol{f})}{\partial \boldsymbol{f} \partial \boldsymbol{f}^T}\right|_{\boldsymbol{f}=\boldsymbol{f}_{\text{MP}}} = \Sigma^{-1} + C \cdot \Lambda$ where 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. The Laplacian approximation of $S(\boldsymbol{f})$ can be simplified as in (4.36). By introducing the Laplacian approximation (4.36) and $\mathcal{Z}_{\boldsymbol{f}} = (2\pi)^{n/2}|\Sigma|^{1/2}$ which is defined as in (4.4) into (4.35), the integral can be approximated as

$$\mathcal{P}(\mathcal{D}|\theta) \approx (2\pi)^{-n/2}|\Sigma|^{-1/2}\mathcal{Z}_S^{-n} S(\boldsymbol{f}_{\text{MP}}) \int \exp\left(-\frac{1}{2}(\boldsymbol{f} - \boldsymbol{f}_{\text{MP}})^T \cdot (\Sigma^{-1} + C \cdot \Lambda) \cdot (\boldsymbol{f} - \boldsymbol{f}_{\text{MP}})\right) d\boldsymbol{f} \quad \text{(E.2)}$$

Let us regard $\boldsymbol{f}$ as random variables with a joint Gaussian distribution $\mathcal{N}\left(\boldsymbol{f}_{\text{MP}}, (\Sigma^{-1} + C \cdot \Lambda)^{-1}\right)$, and then we realize that the integral should be the normalization factor $(2\pi)^{n/2}|\Sigma^{-1} + C \cdot \Lambda|^{-1/2}$. Substituting the normalization factor for the integral in (E.2), we get the equation (4.37).

E.2 The Derivation for Equation (4.39) $\sim$ (4.41)

The negative log evidence $-\ln \mathcal{P}(\mathcal{D}|\theta)$ can be written as

$$\begin{aligned} -\ln \mathcal{P}(\mathcal{D}|\theta) &= S(\boldsymbol{f}_{\text{MP}}) + \frac{1}{2}\ln\left|\mathbf{I} + \frac{C}{2\beta\epsilon}\Sigma_{\text{M}}\right| + n\ln \mathcal{Z}_S \\ &= \frac{1}{2}{\boldsymbol{f}_{\text{MP}}}^T \cdot \Sigma^{-1} \cdot \boldsymbol{f}_{\text{MP}} + C\sum_{i=1}^{n}\ell_{\beta,\epsilon}(y_i - f_{\text{MP}}(x_i)) + \frac{1}{2}\ln\left|\mathbf{I} + \frac{C}{2\beta\epsilon}\Sigma_{\text{M}}\right| + n\ln \mathcal{Z}_S \end{aligned} \tag{E.3}$$

where $\mathcal{Z}_S$ is defined as in (2.32) and Σ_{M} is a sub-matrix of the covariance matrix Σ which do not depend on C and ϵ. Note that $\boldsymbol{f}$ is a function of the hyperparameters.

The derivative of $-\ln \mathcal{P}(\mathcal{D}|\theta)$ with respect to $\ln C$ can be written as:

$$\begin{aligned} \frac{\partial -\ln \mathcal{P}(\mathcal{D}|\theta)}{\partial \ln C} &= \left\{ \frac{\partial S(\boldsymbol{f})}{\partial \boldsymbol{f}^T}\Big|_{\boldsymbol{f}=\boldsymbol{f}_{\text{MP}}} \cdot \frac{\partial \boldsymbol{f}}{\partial C} + \frac{\partial C\sum_{i=1}^{n}\ell_{\beta,\epsilon}(y_i - f_{\text{MP}}(x_i))}{\partial C} + \frac{1}{2}\frac{\partial \ln\left|\mathbf{I} + \frac{C}{2\beta\epsilon}\Sigma_{\text{M}}\right|}{\partial C} + n\frac{\partial \ln \mathcal{Z}_S}{\partial C} \right\} \cdot \frac{\partial C}{\partial \ln C} \\ &= \left\{ \sum_{i=1}^{n}\ell_{\beta,\epsilon}(y_i - f_{\text{MP}}(x_i)) + \frac{1}{2}\text{trace}\left[\left(\mathbf{I} + \frac{C}{2\beta\epsilon}\Sigma_{\text{M}}\right)^{-1} \cdot \frac{\partial(\mathbf{I} + \frac{C}{2\beta\epsilon}\Sigma_{\text{M}})}{\partial C}\right] + \frac{n}{\mathcal{Z}_S}\frac{\partial \mathcal{Z}_S}{\partial C} \right\} \cdot C \end{aligned} \tag{E.4}$$

where $\dfrac{\partial \mathcal{Z}_S}{\partial C} = -\sqrt{\beta\epsilon\pi}C^{-\frac{3}{2}}\text{erf}(\sqrt{C\beta\epsilon}) - 2C^{-2}\exp(-C\beta\epsilon)$. After some simplifications, we will reach the equation (4.39).

Analogously, the derivative of $-\ln \mathcal{P}(\mathcal{D}|\theta)$ with respect to $\ln \epsilon$ can be written as:

$$\begin{aligned} \frac{\partial -\ln \mathcal{P}(\mathcal{D}|\theta)}{\partial \ln \epsilon} &= \left\{ \frac{\partial S(\boldsymbol{f})}{\partial \boldsymbol{f}^T}\Big|_{\boldsymbol{f}=\boldsymbol{f}_{\text{MP}}} \cdot \frac{\partial \boldsymbol{f}}{\partial \epsilon} + \frac{\partial C\sum_{i=1}^{n}\ell_{\beta,\epsilon}(y_i - f_{\text{MP}}(x_i))}{\partial \epsilon} + \frac{1}{2}\frac{\partial \ln\left|\mathbf{I} + \frac{C}{2\beta\epsilon}\Sigma_{\text{M}}\right|}{\partial \epsilon} + n\frac{\partial \ln \mathcal{Z}_S}{\partial \epsilon} \right\} \cdot \frac{\partial \epsilon}{\partial \ln \epsilon} \\ &= \left\{ C\sum_{i=1}^{n}\frac{\partial \ell_{\beta,\epsilon}(y_i - f_{\text{MP}}(x_i))}{\partial \epsilon} + \frac{1}{2}\text{trace}\left[\left(\mathbf{I} + \frac{C}{2\beta\epsilon}\Sigma_{\text{M}}\right)^{-1} \cdot \frac{\partial(\mathbf{I} + \frac{C}{2\beta\epsilon}\Sigma_{\text{M}})}{\partial \epsilon}\right] + \frac{n}{\mathcal{Z}_S}\frac{\partial \mathcal{Z}_S}{\partial \epsilon} \right\} \cdot \epsilon \end{aligned} \tag{E.5}$$

where $\dfrac{\partial \mathcal{Z}_S}{\partial \epsilon} = 2(1-\beta) + \sqrt{\dfrac{\beta\pi}{C\epsilon}}\text{erf}(\sqrt{C\beta\epsilon})$ and

$$\frac{\partial \ell_{\epsilon,\beta}(y_i - f_{\text{MP}}(x_i))}{\partial \epsilon} = \begin{cases} -1 & if \quad y_i - f_{\text{MP}}(x_i) \in \Delta_{C^*} \cup \Delta_C \\ -\dfrac{(y_i - f_{\text{MP}}(x_i))^2 - (1-\beta)^2\epsilon^2}{2\beta\epsilon^2} & if \quad y_i - f_{\text{MP}}(x_i) \in \Delta_{M^*} \cup \Delta_M \\ 0 & if \quad y_i - f_{\text{MP}}(x_i) \in \Delta_0 \end{cases}$$

Being aware of the relationship between the noise and the set associated with the samples, we can simplify (E.5) into the form of (4.40).

As for the derivative with respect to $\ln \kappa'$, we have

$$\begin{aligned} \frac{\partial -\ln \mathcal{P}(\mathcal{D}|\theta)}{\partial \ln \kappa'} &= \left\{ \frac{1}{2}\boldsymbol{f}_{\text{MP}}^T\frac{\partial \Sigma^{-1}}{\partial \kappa'}\boldsymbol{f}_{\text{MP}} + \frac{1}{2}\text{trace}\left[\left(\mathbf{I} + \frac{C}{2\beta\epsilon}\Sigma_{\text{M}}\right)^{-1} \cdot \frac{\partial(\mathbf{I} + \frac{C}{2\beta\epsilon}\Sigma_{\text{M}})}{\partial \kappa'}\right] \right\} \cdot \kappa' \\ &= -\frac{\kappa'}{2}\boldsymbol{f}_{\text{MP}}^T\Sigma^{-1}\frac{\partial \Sigma}{\partial \kappa'}\Sigma^{-1}\boldsymbol{f}_{\text{MP}} + \frac{\kappa'}{2}\text{trace}\left[\left(\frac{2\beta\epsilon}{C}\mathbf{I} + \Sigma_{\text{M}}\right)^{-1} \cdot \frac{\partial \Sigma_{\text{M}}}{\partial \kappa'}\right] \end{aligned} \tag{E.6}$$

where $\Sigma^{-1} \cdot \boldsymbol{f}_{\mathrm{MP}} = (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)$ holds at the MAP estimate. Therefore, by using the Lagrangian multiplier vector $\boldsymbol{\alpha} - \boldsymbol{\alpha}^*$ in (E.6), we can get (4.41). Here the inversion of the full matrix Σ can be avoided, while only the inverse of the sub-matrix Σ_{M} with the size of the number of *off-bound* SVs is required.

E.3 The Derivation for Equation (4.46)

We begin by reformulating (4.45) as

$$\mathcal{P}(f(x)|\mathcal{D}) \propto \exp\left(-\frac{1}{2}\frac{(f(x) - \boldsymbol{f}_{\mathrm{MP}}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k})^2}{Cov(x,x) - \boldsymbol{k}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k}}\right) \cdot \int \exp\left(-\frac{1}{2}\Delta\boldsymbol{f}^T \cdot \mathbf{A} \cdot \Delta\boldsymbol{f} + \mathbf{h}^T \cdot \Delta\boldsymbol{f}\right) d\Delta\boldsymbol{f} \tag{E.7}$$

where $\varsigma^2 = Cov(x,x) - \boldsymbol{k}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k}$, $\mathbf{A} = \frac{1}{\varsigma^2}\Sigma^{-1} \cdot \boldsymbol{k} \cdot \boldsymbol{k}^T \cdot \Sigma^{-1} + \Sigma^{-1} + C \cdot \Lambda$, $\mathbf{h} = \frac{1}{\varsigma^2}(f(x) - \boldsymbol{f}_{\mathrm{MP}}^T \cdot \Sigma^{-1} \cdot \boldsymbol{k}) \cdot \Sigma^{-1} \cdot \boldsymbol{k}$ and $\Delta\boldsymbol{f} = \boldsymbol{f} - \boldsymbol{f}_{\mathrm{MP}}$.

Here $\mathbf{A}$ is a $n \times n$ real symmetric matrix and $\Delta\boldsymbol{f}$ is a n-dimensional column vector, and the integral is over the whole of $\Delta\boldsymbol{f}$ space. Now we focus on the integral that can be evaluated by an explicit expression. In order to evaluate the integral it is convenient to consider the eigenvector equations for $\mathbf{A}$ in the form $\mathbf{A}\mathbf{u}_k = \lambda_k \mathbf{u}_k$. Since $\mathbf{A}$ is real and symmetric, we can find a complete orthonormal set of the eigenvectors that satisfies $\mathbf{u}_k^T \mathbf{u}_l = \delta_{kl}$. We can then expand the vector $\Delta\boldsymbol{f}$ as a linear combination of the eigenvectors $\Delta\boldsymbol{f} = \sum_{k=1}^n \tau_k \mathbf{u}_k$. The integral over the $\Delta\boldsymbol{f}$ space can now be replaced by an integration over the coefficient values $d\tau_1, d\tau_2, \ldots, d\tau_n$. The Jacobian of this change of variables is given by $J = \det(\mathbf{U})$ where the columns of the matrix $\mathbf{U}$ are given by the $\mathbf{u}_k$. Since $\mathbf{U}$ is an orthogonal matrix that satisfies $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, $J^2 = \det(\mathbf{U})^T \det(\mathbf{U}) = \det(\mathbf{U}^T\mathbf{U}) = 1$, and hence $|J| = 1$. Using the orthonormality of the $\mathbf{u}_k$, we have $\Delta\boldsymbol{f}^T \mathbf{A} \Delta\boldsymbol{f} = \sum_{k=1}^n \lambda_k \tau_k^2$. We can also define h_k to be the projections of $\mathbf{h}$ onto the eigenvalues as $h_k = \mathbf{h}^T \mathbf{u}_k$. These lead to a set of decoupled integrals over the τ_k of the form

$$\mathcal{I}_n = \prod_{k=1}^n \int_{-\infty}^{+\infty} \exp\left(-\frac{1}{2}\lambda_k \tau_k^2 + h_k \tau_k\right) d\tau_k$$

The product of the integrals can be evaluated as

$$\mathcal{I}_n = (2\pi)^{n/2} \prod_{k=1}^n \lambda_k^{-1/2} \exp\left(\sum_{i=1}^n \frac{h_k^2}{2\lambda_k}\right) \tag{E.8}$$

We note that the determinant of a matrix is given by the product of its eigenvalues, i.e. $|\mathbf{A}| = \prod_{k=1}^n \lambda_k$, and the inverse $\mathbf{A}^{-1}$ has the same eigenvector of $\mathbf{A}$, but with eigenvalues λ_k^{-1}. Since

$h_k = \mathbf{h}^T\mathbf{u}_k$ and $\mathbf{A}^{-1}\mathbf{u}_k = \lambda_k^{-1}\mathbf{u}_k$, we see that $\mathbf{h}^t\mathbf{A}^{-1}\mathbf{h} = \sum_{k=1}^{n}\frac{h_k^2}{\lambda_k}$. Using these results, we can rewrite (E.8) into the following form:

$$\mathcal{I}_n = (2\pi)^{n/2}|\mathbf{A}|^{-1/2}\exp\left(\frac{1}{2}\mathbf{h}^T\mathbf{A}^{-1}\mathbf{h}\right) \tag{E.9}$$

Using the explicit expression of the integral (E.9), (E.7) can be written as

$$\mathcal{P}(f(x)|\mathcal{D}) \propto \exp\left(-\frac{1}{2}\left(\frac{1}{\varsigma^2} - \frac{1}{\varsigma^4}\boldsymbol{k}^T\Sigma^{-1}\mathbf{A}^{-1}\Sigma^{-1}\boldsymbol{k}\right)\left(f(x) - \boldsymbol{f}_{\mathrm{MP}}^T\cdot\Sigma^{-1}\cdot\boldsymbol{k}\right)^2\right) \tag{E.10}$$

It is easy to see that the mean of the Gaussian distribution is $\boldsymbol{f}_{\mathrm{MP}}^T\cdot\Sigma^{-1}\cdot\boldsymbol{k}$ which is equivalent to $(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T\cdot\boldsymbol{k}$ at the solution of (4.25), and the variance σ_t^2 is $\left(\frac{1}{\varsigma^2} - \frac{1}{\varsigma^4}\boldsymbol{k}^T\Sigma^{-1}\mathbf{A}^{-1}\Sigma^{-1}\boldsymbol{k}\right)^{-1}$.

Now we go ahead to further simplify the variance σ_t^2. Applying the *Woodbury's equality*[1], we obtain that $\sigma_t^2 = \varsigma^2 + \boldsymbol{k}^T\Sigma^{-1}(\Sigma^{-1} + C\cdot\Lambda)^{-1}\Sigma^{-1}\boldsymbol{k}$. The first term in σ_t^2 is the original variance in the Gaussian process prediction, and the last term is the contribution coming from the uncertainty in the function values $\boldsymbol{f}$. Note that only off-bound SVs play roles in the predictive distribution (E.7). The variance of the predictive distribution can be compactly written as

$$\sigma_t^2 = Cov(x,x) - \boldsymbol{k}_{\mathrm{M}}^T\cdot\left(\Sigma_{\mathrm{M}}^{-1} - \Sigma_{\mathrm{M}}^{-1}(\Sigma_{\mathrm{M}}^{-1} + \frac{C}{2\beta\epsilon}\mathbf{I})^{-1}\Sigma_{\mathrm{M}}^{-1}\right)\cdot\boldsymbol{k}_{\mathrm{M}} \tag{E.11}$$

where 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, and $\boldsymbol{k}_{\mathrm{M}}$ is a sub-vector of $\boldsymbol{k}$ obtained by keeping the entries associated with the *off-bound* SVs. Let us apply the *Woodbury's equality* again in the bracket of (E.11), we can finally simplify the variance of the predictive distribution as

$$\sigma_t^2 = Cov(x,x) - \boldsymbol{k}_{\mathrm{M}}^T\cdot\left(\frac{2\beta\epsilon}{C}\mathbf{I} + \Sigma_{\mathrm{M}}\right)^{-1}\cdot\boldsymbol{k}_{\mathrm{M}} \tag{E.12}$$

Another way to simplify σ_t^2 to the form (E.12) is to use block matrix manipulation by noting the sparseness in Λ.

[1] *Woodbury's equality*: Let A and B denote two positive definite matrices related by $\mathrm{A} = \mathrm{B}^{-1} + \mathrm{CDC}^T$ where C and D are two other matrices. The inverse of matrix A is defined by $\mathrm{A}^{-1} = \mathrm{B} - \mathrm{BC}(\mathrm{D}^{-1} + \mathrm{C}^T\mathrm{BC})^{-1}\mathrm{C}^T\mathrm{B}$.

E.4 The Derivation for Equation (5.36)

The negative logarithm of the evidence $-\ln \mathcal{P}(\mathcal{D}|\theta)$ (5.35) can also be equivalently written as

$$\begin{aligned} &-\ln \mathcal{P}(\mathcal{D}|\theta) = S(f_{\mathrm{MP}}) + \frac{1}{2}\ln|\mathbf{I} + \Sigma_{\mathrm{M}} \cdot \Lambda_{\mathrm{M}}| \\ &= \frac{1}{2} {f_{\mathrm{MP}}}^T \cdot \Sigma^{-1} \cdot f_{\mathrm{MP}} + 2\sum_{m \in \mathrm{SVs}} \ln \sec\left(\frac{\pi}{4}(1 - y_{x_m} \cdot f_{\mathrm{MP}}(x_m))\right) + \frac{1}{2}\ln|\mathbf{I} + \Sigma_{\mathrm{M}} \cdot \Lambda_{\mathrm{M}}| \end{aligned} \tag{E.13}$$

where $\mathbf{I}$ is the identity matrix with the size of SVs, $m \in \mathrm{SVs}$ denotes m belongs to the index set of SVs. We note that f_{MP} is dependent on θ. Therefore the derivative of the negative logarithm of the evidence with respect to θ can be written as

$$\begin{aligned} -\frac{\partial \ln P(D|\theta)}{\partial \theta} = \; & \frac{1}{2} f_{\mathrm{MP}}^T \cdot \frac{\partial \Sigma^{-1}}{\partial \theta} \cdot f_{\mathrm{MP}} + \frac{1}{2}\frac{\partial \ln|\mathbf{I} + \Sigma_{\mathrm{M}} \cdot \Lambda_{\mathrm{M}}|}{\partial \theta} \\ & + \sum_{i=1}^{n} \left(\frac{\partial S(f_{\mathrm{MP}})}{\partial f_{\mathrm{MP}}(x_i)} + \frac{1}{2}\frac{\partial \ln|\mathbf{I} + \Sigma_{\mathrm{M}} \cdot \Lambda_{\mathrm{M}}|}{\partial f_{\mathrm{MP}}(x_i)} \right) \cdot \frac{\partial f_{\mathrm{MP}}(x_i)}{\partial \theta} \end{aligned} \tag{E.14}$$

The first two terms at the right of the above equation are usually regarded as the explicit parts for θ, while the last term is the implicit part via f_{MP}.

We notice that Σ_{M} is dependent on θ explicitly, while Λ_{M} is the sub-matrix of Λ which is a diagonal matrix with ii-th elements $\frac{\partial^2 \ell_t(y_{x_i} \cdot f_{\mathrm{MP}}(x_i))}{\partial f_{\mathrm{MP}}^2(x_i)}$, refer to (5.10). Obviously $\frac{\partial S(f_{\mathrm{MP}})}{\partial f_{\mathrm{MP}}(x_i)} = 0$ and $\frac{\partial \ln|\mathbf{I}+\Sigma_{\mathrm{M}} \cdot \Lambda_{\mathrm{M}}|}{\partial f_{\mathrm{MP}}(x_i)} = \mathrm{trace}\left((\mathbf{I} + \Sigma_{\mathrm{M}} \cdot \Lambda_{\mathrm{M}})^{-1} \cdot \Sigma_{\mathrm{M}} \cdot \frac{\partial \Lambda_{\mathrm{M}}}{\partial f_{\mathrm{MP}}(x_i)}\right)$. Only the entry corresponding to the SVs x_m is non-zero in the matrix $\frac{\partial \Lambda_{\mathrm{M}}}{\partial f_{\mathrm{MP}}(x_m)}$, that exactly is $\upsilon_{\mathrm{M}}^m \cdot \Lambda_{\mathrm{M}}^m$. Thus the non-zero term $\frac{\partial \ln|\mathbf{I}+\Sigma_{\mathrm{M}} \cdot \Lambda_{\mathrm{M}}|}{\partial f_{\mathrm{MP}}(x_m)}$ is equal to the element of $\upsilon_{\mathrm{M}}^m \cdot \left((\Lambda_{\mathrm{M}}^{-1} + \Sigma_{\mathrm{M}})^{-1} \cdot \Sigma_{\mathrm{M}}\right)_{mm}$. As for the term $\frac{\partial f_{\mathrm{MP}}(x_i)}{\partial \theta}$, we notice that $f_{\mathrm{MP}} = \Sigma \cdot \upsilon$, refer to (5.19), and the value $\upsilon_i = -\left.\frac{\partial \ell_t(y_{x_i} \cdot f(x_i))}{\partial f(x_i)}\right|_{f(x_i)=f_{\mathrm{MP}}(x_i)}$ is also a function of θ implicitly. Now we get $\frac{\partial f_{\mathrm{MP}}}{\partial \theta} = \frac{\partial \Sigma}{\partial \theta}\upsilon + \Sigma(\frac{\partial \upsilon}{\partial f_{\mathrm{MP}}^T}\frac{\partial f_{\mathrm{MP}}}{\partial \theta})$. Notice that $\frac{\partial \upsilon}{\partial f_{\mathrm{MP}}^T} = -\Lambda$. Therefore we get $(I + \Sigma \cdot \Lambda)\frac{\partial f_{\mathrm{MP}}}{\partial \theta} = \frac{\partial \Sigma}{\partial \theta}\upsilon$, and then

$$\frac{\partial f_{\mathrm{MP}}}{\partial \theta} = (I + \Sigma \cdot \Lambda)^{-1}\frac{\partial \Sigma}{\partial \theta}\upsilon = \Lambda^{-1}(\Lambda^{-1} + \Sigma)^{-1}\frac{\partial \Sigma}{\partial \theta}\upsilon \tag{E.15}$$

Based on these analysis, we can further simplify (E.14) as

$$\begin{aligned} -\frac{\partial \ln P(D|\theta)}{\partial \theta} = \; & -\frac{1}{2} f_{\mathrm{MP}}^T \cdot \Sigma^{-1} \cdot \frac{\partial \Sigma}{\partial \theta} \cdot \Sigma^{-1} \cdot f_{\mathrm{MP}} + \frac{1}{2}\mathrm{trace}\left((\Lambda_{\mathrm{M}}^{-1} + \Sigma_{\mathrm{M}})^{-1} \cdot \frac{\partial \Sigma_{\mathrm{M}}}{\partial \theta}\right) \\ & + \frac{1}{2}\sum_{m \in \mathrm{SVs}} \upsilon_{\mathrm{M}}^m \cdot \left((\Lambda_{\mathrm{M}}^{-1} + \Sigma_{\mathrm{M}})^{-1} \cdot \Sigma_{\mathrm{M}}\right)_{mm} \cdot \left(\Lambda_{\mathrm{M}}^{-1}(\Lambda_{\mathrm{M}}^{-1} + \Sigma_{\mathrm{M}})^{-1}\frac{\partial \Sigma_{\mathrm{M}}}{\partial \theta}\upsilon_{\mathrm{M}}\right)^m \end{aligned} \tag{E.16}$$

Being aware of

$$\frac{\partial - \ln P(D|\theta)}{\partial \ln \theta} = \frac{\partial - \ln P(D|\theta)}{\partial \theta}\frac{\partial \theta}{\ln \theta} = -\frac{\partial \ln P(D|\theta)}{\partial \theta} \cdot \theta$$

and

$$f_{\mathrm{MP}}^{T} \cdot \Sigma^{-1} \cdot \frac{\partial \Sigma}{\partial \theta} \cdot \Sigma^{-1} \cdot f_{\mathrm{MP}} = v_{\mathrm{M}}^{T} \cdot \frac{\partial \Sigma_{\mathrm{M}}}{\partial \theta} \cdot v_{\mathrm{M}}$$

we can reach (5.36) finally.

Appendix F

Noise Generator

Given a random variable u with a uniform distribution in the interval $(0,1)$, we wish to find a function $g(u)$ such that the distribution of the random variable $z = g(u)$ is a specified function $F_z(z)$. We maintain that $g(u)$ is the inverse of the function $u = F_z(z)$: if $z = F_z^{-1}(u)$, then $\mathcal{P}(z \leq z) = F_z(z)$ (refer to Papoulis (1991) for a proof).

Given the probability density function $\mathcal{P}_S(\delta)$ as in (2.31), the corresponding distribution function should be

$$F(\delta) = \begin{cases} \frac{1}{C\mathcal{Z}_S}\exp(C(\delta+\epsilon)) & if \quad \delta \in \Delta_{C^*} \\ \frac{1}{2} - \frac{1}{\mathcal{Z}_S}\left[(1-\beta)\epsilon - \sqrt{\frac{\pi\beta\epsilon}{C}}\mathrm{erf}\left(\sqrt{\frac{C}{4\beta\epsilon}}(\delta+(1-\beta)\epsilon)\right)\right] & if \quad \delta \in \Delta_{M^*} \\ \frac{1}{2} + \frac{\delta}{\mathcal{Z}_S} & if \quad \delta \in \Delta_0 \\ \frac{1}{2} + \frac{1}{\mathcal{Z}_S}\left[(1-\beta)\epsilon + \sqrt{\frac{\pi\beta\epsilon}{C}}\mathrm{erf}\left(\sqrt{\frac{C}{4\beta\epsilon}}(\delta-(1-\beta)\epsilon)\right)\right] & if \quad \delta \in \Delta_M \\ 1.0 - \frac{1}{C\mathcal{Z}_S}\exp(-C(\delta-\epsilon)) & if \quad \delta \in \Delta_C \end{cases} \tag{F.1}$$

Now we solve the inverse problem to sample data in the distribution (F.1): given a uniform distribution of u in the interval $(0,1)$, we let $\delta = F_\delta^{-1}(u)$ so that the distribution of the random variable δ equals to the distribution (F.1).

Appendix G

Convex Programming with Trust Region

Let us consider a general optimization problem, such as $\min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha})$, where $f(\boldsymbol{\alpha})$ is at least C^2 smooth, i.e., the gradient $\frac{\partial f(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}$ and the Hessian matrix $\frac{\partial^2 f(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha} \partial \boldsymbol{\alpha}^T}$ are available. Since the function is smooth, the local minima occur at stationary points, i.e., zeros of the gradient. Newton's method or some variants like the quasi-Newton methods are widely used for finding a zero of the gradient. The line-search approach is usually employed to determine the optimal step length along the descent direction. An alternate approach is based on the observation that quasi-Newton methods model the function by a quadratic approximation around the current point $\boldsymbol{\alpha}^c$. The quadratic is accurate only in a neighborhood of the current point $\boldsymbol{\alpha}^c$, and then the new next point $\boldsymbol{\alpha}^{c+1}$ is now chosen to be an approximate minimizer of the quadratic constrained to be in the region where we trust the approximation (Steihaug, 1983). Let us approximate the original problem around the current point $\boldsymbol{\alpha}^c$ as $f(\boldsymbol{\alpha}) \approx f(\boldsymbol{\alpha}^c) + \triangle \boldsymbol{\alpha}^T \cdot \mathbf{g}_c + \frac{1}{2} \triangle \boldsymbol{\alpha}^T \cdot \mathbf{H}_c \cdot \triangle \boldsymbol{\alpha}$ with $\triangle \boldsymbol{\alpha} = \boldsymbol{\alpha} - \boldsymbol{\alpha}^c$, the gradient $\mathbf{g}_c$ and the Hessian matrix $\mathbf{H}_c$ at $\boldsymbol{\alpha}^c$.

Now we attempt to minimize the functional $f(\boldsymbol{\alpha})$ within the trust region. The consequent optimization problem is

$$\min_{\triangle \boldsymbol{\alpha}} \Phi(\triangle \boldsymbol{\alpha}) = \triangle \boldsymbol{\alpha}^T \cdot \mathbf{g}_c + \frac{1}{2} \triangle \boldsymbol{\alpha}^T \cdot \mathbf{H}_c \cdot \triangle \boldsymbol{\alpha} \tag{G.1}$$

subject to $\triangle \boldsymbol{\alpha}^T \cdot \mathbf{C} \cdot \triangle \boldsymbol{\alpha} \leq \delta$, where δ is a fixed upper bound on the step lengths, and $\mathbf{C}$ is a symmetric and positive definite matrix. A general algorithm using conjugate gradient methods has been proposed by Steihaug (1983) for the solution of (G.1).

In the problem (5.29), we adapt the popular SMO algorithm for classical SVC (Platt, 1999; Keerthi et al., 2001) 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 (5.31) is satisfied. The updating steps in sub-optimization problems turn out to be particular cases of (G.1) with two variables only. We may use multiple Newton's formula or Newton-Raphson steps to update the two Lagrange multipliers till convergence, then we return to SMO to update the variables β^{up} and β^{low}, and check the stopping condition (5.31).

In our particular case of (G.1), it is easy to obtain the unconstrained optimal solution ${\mathbf{H}_c}^{-1} \cdot \mathbf{g}_c$, since $\mathbf{H}_c$ is positive definite and the size is 2×2. If the unconstrained solution is out of the trust region, we resort to Newton's methods as an alternative. The details of the updating 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

$$\begin{bmatrix} \alpha_i^{new} \\ \alpha_j^{new} \end{bmatrix} = \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} - {\mathbf{H}_c}^{-1} \cdot \mathbf{g}_c = \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} - \begin{bmatrix} H_{ii} & H_{ij} \\ H_{ij} & H_{jj} \end{bmatrix}^{-1} \cdot \begin{bmatrix} -y_{x_i} \cdot F_i \\ -y_{x_j} \cdot F_j \end{bmatrix} \tag{G.2}$$

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 (5.30). In the case of $\mathbf{g}_c^T \cdot {\mathbf{H}_c}^{-1} \cdot \mathbf{C} \cdot {\mathbf{H}_c}^{-1} \cdot \mathbf{g}_c > \delta$,[1] we use Newton's formula in place of Newton-Raphson step (G.2), that is

$$\begin{bmatrix} \alpha_i^{new} \\ \alpha_j^{new} \end{bmatrix} = \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} - \gamma \cdot \begin{bmatrix} -y_{x_i} \cdot F_i \\ -y_{x_j} \cdot F_j \end{bmatrix} \tag{G.3}$$

where $\gamma = \min\{\sqrt{\frac{\delta}{\mathbf{g}_c^T \cdot \mathbf{C} \cdot \mathbf{g}_c}}, \frac{\mathbf{g}_c^T \cdot \mathbf{g}_c}{\mathbf{g}_c^T \cdot \mathbf{H}_c \cdot \mathbf{g}_c}\}$. So, before we return to SMO to update the variables β^{up} and β^{low}, and check the stopping condition (5.31), we use (G.2) or (G.3) 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.

[1] We simply use identity matrix as $\mathbf{C}$, and set δ at 1 in this case.

Appendix H

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

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

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 (H.1) can be rewritten as

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

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 (H.2) are

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

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

$$\frac{\pi}{2}\tan\left(\frac{\pi}{4}\xi_i\right) = \alpha_i + \gamma_i, \forall i. \tag{H.3}$$

Notice that the implicit constraint $\xi_i < 2$ has been taken into account automatically in (H.3). Following the analogous arguments in Section 5.3.2, we can derive the *dual* problem as

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

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. Popular SMO algorithm (Platt, 1999; Keerthi et al., 2001) can be adapted to find the solution. The classifier is obtained as $f(x) = \langle\sum_{i=1}^{n} y_{x_i}\alpha_i\phi(x_i)\cdot\phi(x)\rangle + b = \sum_{i=1}^{n} y_{x_i}\alpha_i K(x_i,x) + b$ at the optimal solution of the *dual* problem (H.4), where the bias b can also be obtained. Cross validation is usually used to choose the optimal parameters for the kernel function.

We give the experimental results of TSVC on U.S. Postal Service data set of handwritten digits (USPS) via 5-fold cross validation. USPS is a large scale data set with 7291 training samples and 2007 test samples of 16×16 grey value images, where grey values of pixels are scaled to lie in $[-1, +1]$.[1] It is a 10-class classification problem. The 10-class classifier could be constructed by 10 one-versus-rest (1-v-r) classifiers. The i-th classifier will be trained with all of the samples in the i-th class with positive labels, and all other samples with negative labels. The final output is decided as the class that corresponds to the 1-v-r classifier with the highest output value. Platt et al. (2000) trained 10 1-v-r classical SVCs with Gaussian kernel in this way, and reported that the error rate is 4.7%, where the model parameters were determined by cross validation. Strictly in the same way, we train 10 1-v-r TSVCs with Gaussian kernel where the hyperparameter is determined by cross validation too. Their individual training results are reported in Table H.1. The final testing error rate of the ten-class TSVC is 4.58%. Notice that the CPU time consumed by one TSVC training is around 300 seconds.[2]

[1] It is available from http://www.kernel-machines.org/data.html

[2] In the program implementation, we did not encode the sparseness in dot product or cache the kernel matrix.

Table H.1: Training results of the 10 1-v-r TSVCs with Gaussian kernel on the USPS handwriting digits, where $\kappa_0 = 15$ and $\kappa = 0.02$ determined by cross validation. Time denotes the CPU time in seconds consumed by one TSVC training; Training Error denotes the number of training error; Test Error denotes the number of misclassified samples in test and Test Error Rate denotes the test error in percentage of these binary classifiers.

Digit	Training Error	SVs	Time	Test Error	Test Error Rate
0	0	611	240.0	9	0.448
1	1	176	131.1	9	0.448
2	0	842	398.7	30	1.495
3	0	734	354.9	20	0.997
4	1	755	456.6	31	1.545
5	0	880	425.3	22	1.096
6	0	571	284.5	16	0.797
7	0	502	270.3	14	0.698
8	0	834	418.4	27	1.345
9	0	624	335.4	17	0.847