Bayesian Locally Weighted Online Learning

Narayanan U. Edakunni

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2009

Abstract

Locally weighted regression is a non-parametric technique of regression that is capable of coping with non-stationarity of the input distribution. Online algorithms like Receptive Field Weighted Regression and Locally Weighted Projection Regression use a sparse representation of the locally weighted model to approximate a target function, resulting in an efficient learning algorithm. However, these algorithms are fairly sensitive to parameter initializations and have multiple open learning parameters that are usually set using some insights of the problem and local heuristics. In this thesis, we attempt to alleviate these problems by using a probabilistic formulation of locally weighted regression followed by a principled Bayesian inference of the parameters.

In the Randomly Varying Coefficient (RVC) model developed in this thesis, locally weighted regression is set up as an ensemble of regression experts that provide a local linear approximation to the target function. We train the individual experts independently and then combine their predictions using a Product of Experts formalism. Independent training of experts allows us to adapt the complexity of the regression model dynamically while learning in an online fashion. The local experts themselves are modeled using a hierarchical Bayesian probability distribution with Variational Bayesian Expectation Maximization steps to learn the posterior distributions over the parameters. The Bayesian modeling of the local experts leads to an inference procedure that is fairly insensitive to parameter initializations and avoids problems like overfitting. We further exploit the Bayesian inference procedure to derive efficient online update rules for the parameters. Learning in the regression setting is also extended to handle a classification task by making use of a logistic regression to model discrete class labels.

The main contribution of the thesis is a spatially localised online learning algorithm set up in a probabilistic framework with principled Bayesian inference rule for the parameters of the model that learns local models completely independent of each other, uses only local information and adapts the local model complexity in a data driven fashion. This thesis, for the first time, brings together the computational efficiency and the adaptability of 'non-competitive' locally weighted learning schemes and the modelling guarantees of the Bayesian formulation.

Acknowledgements

I would like to thank Dr. Sethu Vijayakumar for all his insights and suggestions. I especially thank him for his unending enthusiasm which propelled me whenever I was down. A special thanks to Dr. Tim Kovacs and Dr. Gavin Brown for being a generous employer, allowing me to work on my PhD even while providing me employment on the project headed by them.

I would also like to thank all my colleagues in the SLMC lab including Graham McNeill, Giorgos Petkos, Timothy Hospedales, Adrian Haith, Matthew Howard, Sebastian Bitzer and Djordje Mitrovic who supported me technically and personally.

Special thanks to my friends Rowena and Jovito who provided a home away from home whenever I needed one and to my caring family who stood firm behind me and encouraged me all through the years.

Finally my big thanks to my wife Kairali for her infinite patience and support that ultimately helped me complete this thesis.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Narayanan U. Edakunni*)

To Raman Unny and Bhama Unny

Table of Contents

Notations

$\mathbf{A}$	matrix
$\mathbf{A}^T$	transpose of matrix $\mathbf{A}$
$trace(\mathbf{A})$	trace of matrix $\mathbf{A}$
$\mathbf{a}$	vector
a	scalar
$\mathbf{A}(i,j)$	(i,j).th entry of matrix $\mathbf{A}$
$diag(.)$	diagonal matrix constructed from the argument
$\mathcal{N}(.,.)$	Normal distribution
$\mathcal{G}(.,.)$	Gamma distribution
$I\mathcal{G}(.,.)$	Inverse Gamma distribution
$\langle . \rangle_q$	expectation w.r.t q
$f()$	function
$sgn(.)$	sign of the argument
$(.)_+$	function that takes value zero if the argument is less than zero otherwise takes the value of the argument itself
$\|.\|_1$	l_1 norm of the argument
$\nabla\mathbf{A}$	gradient of $\mathbf{A}$
$\nabla\nabla\mathbf{A}$	hessian of $\mathbf{A}$

Standard probability distributions

Normal	$P(\theta) = \frac{1}{\sqrt{2\pi\sigma}} exp\left(-\frac{1}{2\sigma^2}(\theta-\mu)^2\right)$
Multivariate normal	$P(\theta) = (2\pi)^{-d/2}\lvert\Sigma\rvert^{-1/2} exp\left(-\frac{1}{2}(\theta-\mu)^T\Sigma^{-1}(\theta-\mu)\right)$
Gamma	$P(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\theta^{\alpha-1} exp(-\beta\theta)$
Inverse-gamma	$P(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\theta^{-(\alpha+1)} exp(-\beta/\theta)$
Laplacian	$P(\theta) = \frac{1}{2\sigma^2} exp(-\frac{\lvert\theta-\mu\rvert}{\sigma^2})$

Chapter 1

Introduction

Recent progress in information systems has seen an explosion of data that needs to be processed. The data could typically consist of values of certain variables in the real world and the task would be to infer the relation between these different variables. Different learning systems have been developed to accomplish this task by processing the values observed for these variables. One of the common learning scenario is the *supervised* learning where the task involves deducing the relation between a set of *input* variables and a *response* variable. Various supervised learning algorithms are tailor made to handle different settings of learning depending on the nature of the data and the application for which it is used. In this thesis, we are interested in developing a learning algorithm that has the following characteristics :

1. *Learn from continuous noisy response* : The learning algorithm must be able to infer the mapping f from a multivariate input variable $\mathbf{x}$ to a continuous response variable y given observations of the input variable $\mathbf{x}_1 \dots \mathbf{x}_N$ and the corresponding noisy observations of the response $y_1 \dots y_N$. The value of the observed response can then be modeled as :

   $$y = f(\mathbf{x}) + \varepsilon$$

   where ε is the random variable corresponding to an independent Gaussian noise. We also assume that the mapping f is deterministic and does not change with time.

2. *Large amounts of continually arriving data* : The training data is assumed to be produced continually and the learning algorithm must be capable of dealing with the stream of data - typical scenario pre-empt storing and batch processing. A good example of such a situation is learning the dynamics model of an

anthropomorphic robot (Vijayakumar et al., 2002) from movement data. The intrinsic dynamics of the robot is represented by the mapping between the command (torque) and the desired action (joint angle, joint velocity), and needs to be learnt in tandem with the execution of the command itself. This requires the algorithm to be capable of learning from its experiences, processing data points as they arrive and then discarding them. This paradigm of learning is termed as *online* learning. Large amounts of data also ensures that we do not have to worry about finite-sample effects during online learning.

3. *Computational efficiency and real-time applicability* : In order to implement incremental learning in real time, the learning algorithm should have minimal time complexity.

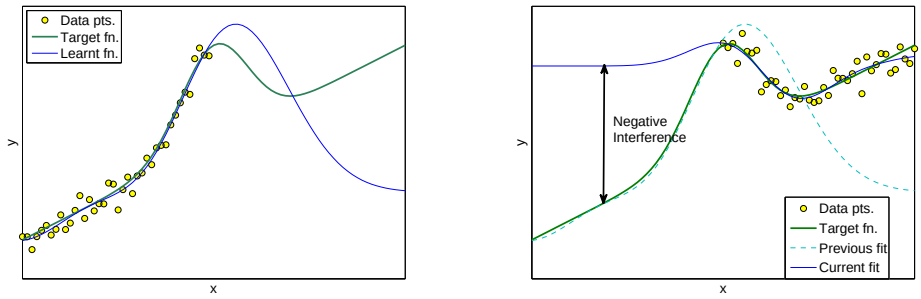
4. *Automatic structure determination* : We assume that there is minimal or no prior knowledge about the complexity of the function f that we are approximating. This precludes the use of any parametric representation of the function - potential methods fall under the *non-parametric* estimation technique, with the model structure being learned from the data.

5. *Non-stationary input distribution* : When a learning system is trained using a stream of data points, the sampling distribution of the input could change with time. Learning in new regions of space can then interfere with the previously learnt fit for the function. This phenomenon is often termed as *negative interference* (Schaal & Atkeson, 1998). In this thesis we are interested in negative interference due to a change in the input distribution and not with a change in the functional relation between the input and the output. To make the distinction clear, we borrow the explanation from (Schaal & Atkeson, 1998) - If we base our estimation of a function f on the minimum squared error criterion, then the estimate $\hat{f}$ would be obtained by minimizing -

$$\int_{-\infty}^{\infty} ||y - \hat{f}(x)||^2 p(x,y) dxdy = \int_{-\infty}^{\infty} ||y - \hat{f}(x)||^2 p(y|x) p(x) dxdy$$

The estimate $\hat{f}$ will thus depend on the input distribution $p(x)$ for finite number of samples. If $p(x)$ changes it can lead to a change in $\hat{f}$ and result in negative interference.

Negative interference is illustrated in Fig. 1.1 where the function approximation learnt in Fig. 1.1(a) is forgotten after learning from data points in a different



(a) Initial function fit

(b) Function fit after a change in the input distribution

Figure 1.1: A schematic illustration of negative interference - (b) illustrates the forgetting of the initial learning in (a) due to a change in the input data distribution.

region of space in Fig. 1.1(b). We aim to minimize the effect of negative interference in methods developed here.

6. *Minimal number of open parameters* : There should be minimal number of open parameters for the learning system so that manual tuning can be avoided.

We now motivate the learning methodology adopted in this thesis by reviewing some of the related algorithms that satisfy a few of the criteria listed above but have other deficiencies that makes it unsuitable for the purpose.

1.1 Non-parametric regression

Non-parametric learning can be used when we lack a definite prior knowledge about the model structure of the underlying generative process. In non-parametric learning, we make an assumption that properties within the neighbourhood of an input point are related in a particular smooth manner. For regression, the assumption could be that the points within a neighbourhood share a specific parametric form for the function (piecewise polynomial smoothers) and for non-parametric classification it could be that the point of interest belongs to the same class as its neighbours (k-nearest neighbour classifier). We concentrate for now on regression but the arguments for regression carries over to classification as well.

A non-parametric regression commonly uses local averaging to estimate the function at a given point. More formally, the estimate of the function $f(\mathbf{x}_c)$ at input point

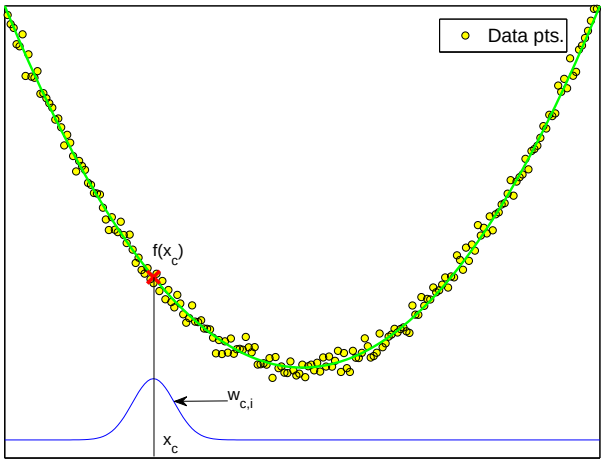


Figure 1.2: Illustration of kernel regression

$\mathbf{x}_c$ is a weighted sum of the training responses given by :

$$f(\mathbf{x}_c) = \sum_{i=1}^{N} w_{c,i} y_i \tag{1.1}$$

where $w_{c,i}$ is the weight provided to each of the training response. The form of non-parametric estimate in eq. (1.1) is also called a *linear smooth* (Hastie & Tibshirani., 1990; Loader, 1999a) because the response at a test point is estimated as a linear combination of the training responses. As illustrated in Fig. 1.2, the weights $w_{c,i}$ in eq. (1.1) are usually chosen such that the responses y_i with the corresponding input points $\mathbf{x}_i$ lying close to $\mathbf{x}_c$ get higher weights while distant points get lesser weights.

There are various methods for non-parametric regression including Kernel smoothing (Nadaraya, 1964; Watson, 1964; Gasser et al., 1991), Orthogonal series estimators (Szegö, 1992), Spline smoothing (Silverman, 1984), Gaussian process regression (Rasmussen & Williams, 2006) and Local polynomial regression (Loader, 1999a). However as explained earlier, we are interested in a spatially localised non-parametric learning algorithm. Locally weighted polynomial regression is one such algorithm for non-parametric regression. Polynomial regression smoothing has many advantages over the other methods of non-parametric regression (Hastie & Loader, 1993; Jones et al., 1994) including simple interpretation and efficient inference. Furthermore, it can also be shown (Härdle, 1994) that kernel smoothers is just a special case of local polynomial regression.

1.1.1 Locally weighted polynomial regression

The first use of localised polynomial regression was in (Gram, 1883). Other early independent developments in the field of local polynomial fitting and smoothing include (De Forest, 1873; De Forest, 1874; Woolhouse, 1870; Spencer, 1904) and is reviewed in (Cleveland & Loader, 1995).

A local polynomial regression assumes that a non-linear function can be approximated locally by a polynomial fit. For instance, a non-parametric local *linear univariate* regression assumes a linear parametric form for the function f within a local region centered around x_c and is given by :

$$f(x_i) \approx \beta_0 + (x_i - x_c)\beta_1$$

Locality is modeled by a weighting function that gives different weights to data points around x_c. Defining $\phi((x_i - x_c)/h)$ to be the weighting function with *bandwidth* h, the estimate $\hat{\beta}$ for the regression coefficient $\beta \equiv [\beta_1 \ \beta_0]^T$ for a local linear fit is obtained by minimizing the weighted squared error loss function :

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}}(\sum_i \phi((x_i - x_c)/h)(y_i - \beta^T \mathbf{x}_i)^2) \tag{1.2}$$

where $\mathbf{x}_i \equiv [(x_i - x_c) \ 1]^T$ and $\hat{\beta} \equiv [\hat{\beta}_1 \ \hat{\beta}_0]^T$. To provide a local estimate, the weighting function $\phi((x_i - x_c)/h)$ is chosen to be symmetric around x_c and decreasing with $|x_i - x_c|$. The bandwidth parameter (also called the smoothing parameter) h modulates the extent of locality. The estimate $\hat{\beta}$ in eq. (1.2) can be written down in a matrix form as :

$$\hat{\beta} = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y} \tag{1.3}$$

where, $\mathbf{W} = diag(\phi((x_i - x_c)/h))$, $\mathbf{X} \equiv [\mathbf{x}_1 \ldots \mathbf{x}_N]$ and $\mathbf{y} \equiv [y_1 \ldots y_N]^T$. The estimate for the fit at the point x_c is then given by $\hat{\beta}_0$ which can be written down using eq. (1.3) as :

$$\begin{aligned} \hat{\beta}_0 &= [0 \ 1]^T (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y} \\ &= \mathbf{w}^T \mathbf{y} \end{aligned} \tag{1.4}$$

where $\mathbf{w} = [0 \ 1]^T (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W}$. It can be seen that eq. (1.4) has the same form as eq. (1.1), thus demonstrating that local polynomial regression is a special form of linear smoothing.

After having examined the estimation of the fit for a locally weighted regression, we now examine the role of the bandwidth parameter in determining the fit of the local model.

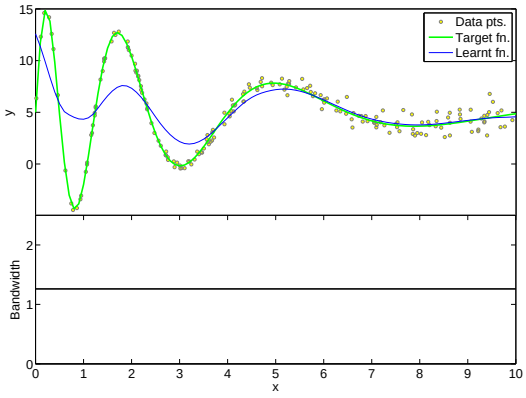
Bandwidth and its estimation

Bandwidth plays a major role in determining the smoothness of the estimate. For a given fixed bandwidth parameter, minimizing the squared error given by eq. (1.2) would give the local linear fit. The fit learnt would be different for different values of bandwidth. As the bandwidth increases, the neighborhood increases and the estimate is smooth and approaches a global parametric fit as $h \rightarrow \infty$. In contrast as the bandwidth decreases the estimate tends to be undersmoothed and in the limit of $h \rightarrow 0$ the estimate is the value of the response itself. A smaller bandwidth thus implies less bias but higher variance of the estimator and vice versa. Hence, it is important to obtain a correct estimate of the bandwidth by balancing the bias against the variance. There are different ways to parametrise a bandwidth in a local regression model - a single bandwidth could be used for all the local models or different bandwidths could be used in different regions of the input space.

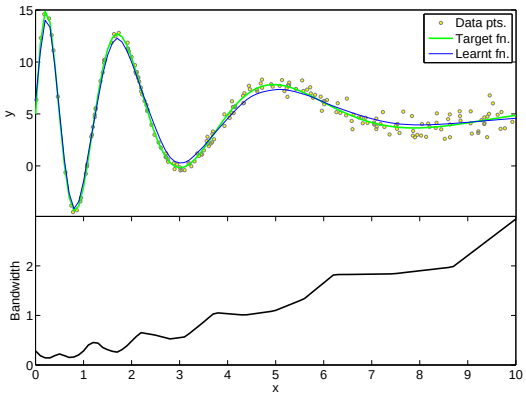
Constant bandwidth is used to model a weighting function that has the same extent of locality across the entire range of the input space. This form of weighting function is easy to interpret but its capacity to model functions is limited to simple forms. It is insufficient to model functions with varying spatial complexity like the one in Fig. 1.3(a). It can be seen from the figure that a constant bandwidth provides a good fit for the linear part of the function but is incapable of modeling the non-linear region of the function with a high bias in this region.

Varying bandwidth is the alternative to this approach where we have different bandwidths at different regions of the input space. The function illustrated in Fig. 1.3(a) is now modeled using a varying bandwidth and is shown in Fig. 1.3(b). The approximation of the function can be seen to be better when a varying bandwidth is used. The bandwidths used in different regions of the input space is shown in the bottom pane of Fig. 1.3(b). It can be seen that larger values of bandwidths are used to model the linear region of the function while smaller values are used to model the more non-linear parts of the function. This results in a fit that adapts to the varying spatial complexities of the function.

There have been different methods proposed for the estimation of bandwidth for non-parametric regression and broadly falls into two categories - *classical* and *plug-in*. The classical methods of estimation of bandwidth are extensions of model selection methods in parametric statistics. These include cross validation (Allen, 1974), Mallow's CP criterion (Mallows, 1973), and Akaike information criterion (Akaike, 1974).



(a) Constant bandwidth fit



(b) Varying bandwidth fit

Figure 1.3: Comparison of local linear fits using a constant versus varying bandwidth. The toy function has a spatially varying complexity. The bandwidths were computed using the LOCFIT software (Loader, 1999a)

In the plug-in approach to bandwidth selection, an analytical expression for the asymptotic mean integrated squared error(Jones et al., 1996) is derived which is then minimized to obtain the optimal bandwidth. The expression derived for the optimal bandwidth however, contains terms of unknowns like the second derivative of the target function and different plug-in methods try to estimate these unknown expressions using the different approximations. Accordingly there have been a variety of plug-in methods which started with (Woodrofe, 1970) and further developed in (Gasser et al., 1991) and (Ruppert et al., 1995). Different plug-in methods have been reviewed in (Fan & Gijbels, 1996) and the comparison of classical methods with the plug-in methods of bandwidth estimation can be found in (Loader, 1999b). It must be noted that all these

methods perform localised non-parametric regression and differ only in their estimates.

From the discussion in this section we can conclude that localised non-parametric regression circumvents the problem of negative interference by localizing the interference using locally weighted learning routine and adapting its model complexity in a data-driven fashion. However, the non-parametric smoother uses a memory based *lazy evaluation* strategy wherein the smoothing algorithm stores away all the training data points and uses a weighted smooth of the training responses (refer eq. (1.1)) to compute the prediction at a new test point. This results in an increased space complexity of the trained model along with an increase in the time complexity for each prediction thus making it unconducive for incremental learning. The solution to this problem lies in *constructively* and *incrementally* building up a representation of the target function by using local models centered at only a *subset* of training points in contrast to the memory based approach of lazy evaluation. One such class of online learning algorithms is the locally weighted regression algorithms as represented by Receptive Field Weighted Regression(RFWR)(Schaal & Atkeson, 1998) and Locally Weighted Projection Regression(LWPR)(Vijayakumar et al., 2005).

1.2 Online locally weighted regression

In this section we look at a popular method of online regression which uses local linear models to approximate a non-linear function and is able to dynamically adapt the complexity of the approximating function by adding local models during learning. The initial version of the algorithm was known as Receptive Field Weighted Regression(RFWR) (Schaal & Atkeson, 1998) which used locally weighted linear models to approximate the function. Scalability of RFWR was improved with the addition of dimensionality reduction of the input space resulting in an algorithm called Locally Weighted Projection Regression(LWPR)(Vijayakumar et al., 2005). In this section, we review the learning procedure formulated in RFWR and briefly describe its extension to higher dimensions in the form of LWPR. As with all locally weighted regression algorithms RFWR uses a weighted error criteria to learn the parameters of the model. The loss function is a form of least squared cross-validation given by :

$$\mathcal{J} = \frac{\sum_{i=1}^{N} (y_i - \hat{\beta}_{-i}^{T} \mathbf{x}_i)^2 \phi(\mathbf{x}_i)}{W} \tag{1.5}$$

where $\hat{\beta}_{-i}$ is the estimate of the regression coefficient estimated using all of the data except the i^{th} point,

$$\phi(\mathbf{x}_i) = exp(-\mathbf{x}_i^T \mathbf{D} \mathbf{x}_i) \tag{1.6}$$

is the weight function with $\mathbf{D}$ being the inverse bandwidth matrix and $W = \sum_i \phi(\mathbf{x}_i)$. The evaluation of the N-fold cross-validation error as represented by eq. (1.5) is computationally expensive since it requires the inference of the regression coefficient N times and the subsequent optimization of $\mathcal{J}$ to infer the bandwidth matrix $\mathbf{D}$. Using Sherman-Morrison-Woodbury theorem it is possible to express eq. (1.5) in terms of the regression coefficient inferred using the entire dataset as :

$$\mathcal{J} = \frac{1}{W} \sum_{i=1}^{N} \frac{(y_i - \hat{\beta}^T \mathbf{x}_i)^2 \phi(\mathbf{x}_i)}{\left(1 - \phi(\mathbf{x}_i) \mathbf{x}_i^T \mathbf{P} \mathbf{x}_i\right)^2} \quad \text{where} \quad \mathbf{P} = \left(\mathbf{X}^T \mathbf{S} \mathbf{X}\right)^{-1} \tag{1.7}$$

where $\mathbf{X} \equiv (\mathbf{x}_1 \ldots \mathbf{x}_N)^T$ and $\mathbf{S} \equiv diag(\phi(\mathbf{x}_1) \ldots \phi(\mathbf{x}_N))$. Minimizing the error criterion given by eq. (1.7) leads to a consistent model - a model whose bias decreases with the number of training data points, but the downside is that the locality also shrinks thus requiring more number of local models to approximate the function (Schaal & Atkeson, 1998). To avoid this, a penalty term is introduced in the loss :

$$\mathcal{J} = \frac{1}{W} \sum_{i=1}^{N} \frac{(y_i - \hat{\beta}^T \mathbf{x}_i)^2 \phi(\mathbf{x}_i)}{\left(1 - \phi(\mathbf{x}_i) \mathbf{x}_i^T \mathbf{P} \mathbf{x}_i\right)^2} + \gamma \sum_{i,j=1}^{d} \mathbf{D}_{ij}^2 \tag{1.8}$$

where the second part of the equation stands for the penalty for a small bandwidth. The *penalty factor* γ controls the relative magnitude of the penalty and in turn influences the smoothness of the fit.

To obtain an online algorithm we need to optimize the objective given by eq. (1.8) incrementally and obtain the updates for learning the bandwidth matrix $\mathbf{D}$. A gradient descent update would be given by :

$$\mathbf{M}^{n+1} = \mathbf{M}^n - \alpha \frac{\partial \mathcal{J}}{\partial \mathbf{M}} \tag{1.9}$$

where $\mathbf{M}$ is an upper triangular matrix such that $\mathbf{D}$ (given by $\mathbf{D} = \mathbf{M}^T \mathbf{M}$) is guaranteed to be positive definite. The optimization of $\mathcal{J}$ using eq. (1.9) can be turned into an incremental update by approximating the gradient $\frac{\partial \mathcal{J}}{\partial \mathbf{M}}$ using a novel stochastic approximation which unlike conventional stochastic gradient descent, uses a memory trace to maintain a history of the sufficient statistics of the data and uses these to update the parameters. This results in a more stable procedure for learning.

We started out by motivating the online learning algorithm as a means of introducing sparsity to the otherwise lazy evaluation methods of localised learning. RFWR

Algorithm 1 Receptive Field Weighted Regression

Input: Training point $\mathbf{x}, y$
for k = 1 to #local models **do**
  Calculate the weight $\phi_k = \exp\left(-(\mathbf{x}-\mathbf{x}_k)^T \mathbf{D}_k (\mathbf{x}-\mathbf{x}_k)\right)$
  Update the bandwidth using eq. (1.9)
end for
if $\phi_k < \phi_{gen} \forall_{k=1...\#\text{local models}}$ **then**
  add a new local model with $\mathbf{x}_c = \mathbf{x}$
end if{add a local model if ϕ is less than a threshold ϕ_{gen}}
if $\exists_{i \neq j, i,j \in \{1...\#\text{local models}\}} \phi_i, \phi_j > \phi_{prune}$ **then**
  remove the local model with the larger $|\mathbf{D}|$
end if{remove a local model if there are more than one local model with weight greater than ϕ_{prune}}

achieves sparsity by adding local models only at points in the input region where the accuracy of the function approximation is low. RFWR then combines the predictions of these local models using a weighted average, to output the prediction for a previously unseen input point $\mathbf{x}_q$:

$$y_q = \frac{\sum_{k=1}^{M} \phi_k(\mathbf{x}_q) y_{q,k}}{\sum_{k=1}^{M} \phi_k(\mathbf{x}_q)}$$

where $y_{q,k}$ is the prediction of the k^{th} local model and y_q is the combined prediction. The basic form of RFWR algorithm is summarized by Algorithm 1.

One of the main drawbacks of RFWR and in general locally weighted learning is the curse of dimensionality that manifests in the form of sparsity of data in the high dimension space. As the dimensionality of the input space increases the number of local models required for accurate approximation increases exponentially. Locally weighted projection regression (LWPR) is a modification of RFWR geared towards solving this problem. It combines the robustness of partial least squares (PLS) regression with the locally weighted approach of RFWR to provide an incremental regression that uses a projected lower dimensional space to perform the local regression. The interesting aspect of LWPR learning is that dimensionality reduction and regression are carried out simultaneously in an incremental fashion.

While RFWR/LWPR achieves our aims of spatially localised non-parametric online regression with data dependent adaptation of model complexity, one of its main drawbacks is that it introduces a number of parameters that needs to be tuned for the

gradient descent to find a reasonable solution for the bandwidth. These parameters include the learning rate α (eq. (1.9)), initialization of the bandwidth matrix $\mathbf{D}$ (eq. (1.6)) and the penalty factor γ (eq. (1.8)). The learning rate controls the rate at which the gradient is followed - a slower learning rate results in a highly damped slow converging $\mathcal{J}$. The penalty term models the prior about the smoothness of the target function. Higher the penalty, smoother the function. Disadvantage of having all these open parameters is that it becomes difficult to assign reasonable values to these parameters. Also, given two different learning models with different parameter settings it becomes difficult to select the model that is the most suitable. We can use the squared error of the response as an indicator but then it would need a separate cross validation data to select the model without overfitting.

The ideal solution to avoid tuning of parameters would be to design an optimization function that is convex. Optimizing the convex objective function with respect to the parameters would lead to a unique solution for the parameters irrespective of the initialization. This however is difficult for the current setting of a locally weighted learning where finding such a convex objective function is inherently difficult because there can be different configurations of the local models that provide equivalent solutions. An alternative philosophy is to formulate a probabilistic model for the locally weighted regression and express our prior belief over the parameters as prior probabilities. Given our priors about the parameters it is then possible to combine the beliefs with the evidence obtained from the data to infer the posterior probability over the parameters using the Bayes rule. The uncertainty in the estimation of the parameters is reflected by the posterior distribution which in turn contributes to the overall uncertainty in the prediction of the response. This is useful when we need to combine independent local models having their own levels of confidence into a robust prediction. Furthermore, a Bayesian model selection allows the complexity of the model to be integrated into the selection process along with the fit over the data (MacKay, 1992) thus avoiding the problem of overfitting. The discussion in this section thus motivates the need for a Bayesian probabilistic formulation of a locally linear regression that avoids the need to tune open parameters and have simple yet robust model selection capabilities. One possible candidate for such a learning algorithm is the mixture of experts.

1.3 Mixture of Experts

Mixture of local experts model for regression is an example of a probabilistic formulation of a locally weighted regression. The earliest mixture model used for regression has been (Xu et al., 1995). This work was extended to a Bayesian formulation in (Waterhouse et al., 1996; Bishop & Svensèn, 2003). Next we describe some details of the original version of mixture model as formulated in (Xu et al., 1995).

A mixture of experts model consists of two components - one, a probabilistic regression model that corresponds to the local fit and the other, a region of locality as represented by a probability distribution over the input region. The former is usually termed as the *expert* model while the latter is known as the *gating* function.

A mixture model can be formulated by expressing our belief about the process in which the data was generated. We start with a multinomial random variable z which can take values in $\{1 \ldots M\}$ where M is the number of local models. The probability that the tuple of $\mathbf{x}_i, y_i$ is generated from the j^{th} model is given by $P(\mathbf{x}_i, y_i | z_i = j)$. The joint probability $P(\mathbf{x}_i, y_i | z_i = j)$ can in turn be factorized as $P(\mathbf{x}_i, y_i | z = j) = P(y_i | \mathbf{x}_i, z_i = j) P(\mathbf{x}_i | z_i = j)$. Hence if the prior probability of z_i taking the value j is $P(z_i = j)$, the joint probability is given by the factorization :

$$P(y_i, \mathbf{x}_i, z_i) = P(y_i | \mathbf{x}_i, z_i) P(\mathbf{x}_i | z_i) P(z_i)$$

and is denoted pictorially by the graphical model in Fig. 1.4. The graphical model illustrates the probabilistic dependency between the various random variables of the model. The probabilistic model can be better understood by marginalizing out the hidden variable z to obtain the joint distribution of the input and the response variables :

$$P(y_i, \mathbf{x}_i) = \sum_{j=1}^{M} P(y_i | \mathbf{x}_i, z_i = j) P(\mathbf{x}_i | z_i = j) P(z_i = j)$$

which in turn can be rewritten as a conditional probability as :

$$P(y_i | \mathbf{x}_i) = \frac{\sum_{j=1}^{M} P(y_i | \mathbf{x}_i, z_i = j) P(\mathbf{x}_i | z_i = j) P(z_i = j)}{\sum_{j=1}^{M} P(\mathbf{x}_i | z_i = j) P(z_i = j)} \tag{1.10}$$

It can be seen from eq. (1.10) that the global fit of a function is defined as a set of spatially local fits. The *fit* part of the localised regression corresponds to $P(y_i | \mathbf{x}_i, z_i = j)$ which for linear regression is defined as a Gaussian :

$$P(y_i | \mathbf{x}_i, z_i = j) = \mathcal{N}(y_i; \beta_j^T \mathbf{x}_i, \sigma_j^2)$$

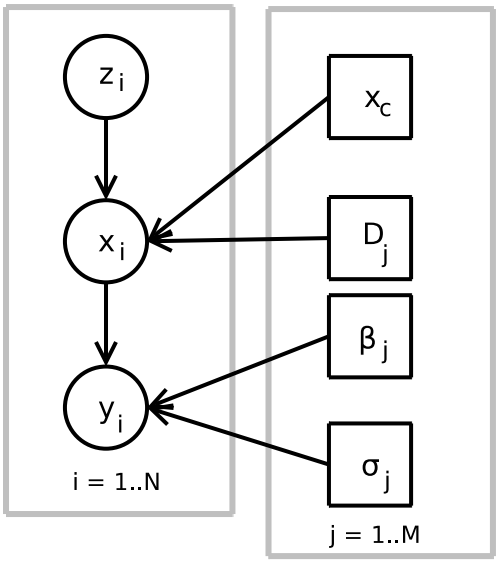


Figure 1.4: Graphical model for a mixture of regression experts

where β_j is the regression coefficient for the j^{th} model and σ_j^2 its noise variance. On the other hand *locality* is defined by $P(\mathbf{x}_i|z_i = j)P(z_i = j)$ where the conditional distribution of the input variable is often modeled by a Gaussian :

$$P(\mathbf{x}_i|z_i = j) = \mathcal{N}(\mathbf{x}_i; \mathbf{x}_j, \mathbf{D}_j)$$

where $\mathbf{x}_c$ is the center of the local model and $\mathbf{D}_j$ is the bandwidth matrix of the j^{th} local model. The maximum likelihood estimates for the parameters - $\beta_j, \sigma_j, \mathbf{x}_c, \mathbf{D}_j$ and $P(z = j)$ can be obtained using an expectation maximization (EM) procedure by treating z as a hidden variable. The predictive distribution for a query point $\mathbf{x}_q$ can then be obtained using the conditional distribution defined by eq. (1.10) and can be written down as :

$$P(y_q|\mathbf{x}_q) = \sum_{j=1}^{M} P(y_q|\mathbf{x}_q, z_q = j) w_{q,j}$$

where :

$$w_{q,j} = \frac{P(\mathbf{x}_q|z_q = j)P(z_q = j)}{\sum_{j=1}^{M} P(\mathbf{x}_q|z_q = j)P(z_q = j)}$$

such that $\sum_j w_{q,j} = 1$. This further implies that the mean prediction is given by :

$$\mathbb{E}(y_q|x_q) = \sum_j w_{q,j} \mathbb{E}(y_q|\mathbf{x}_q, z_q = j)$$

which is a convex combination of the predictions of the individual model. An illustration of linear fits learnt by a mixture of experts along with the responsibilities on a toy example is shown in Fig. 1.5.

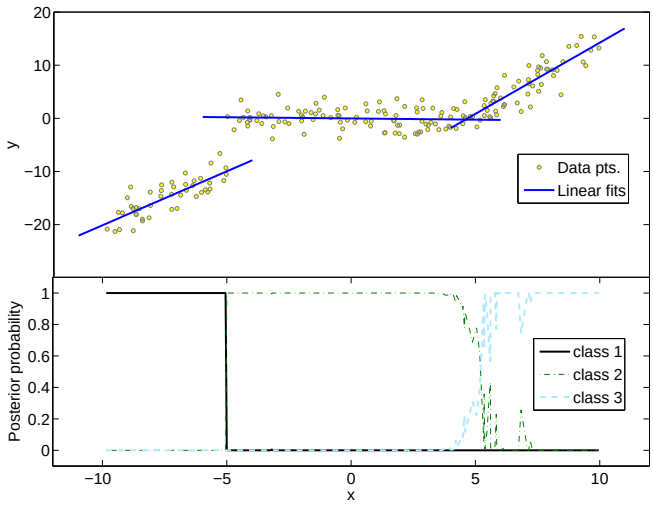
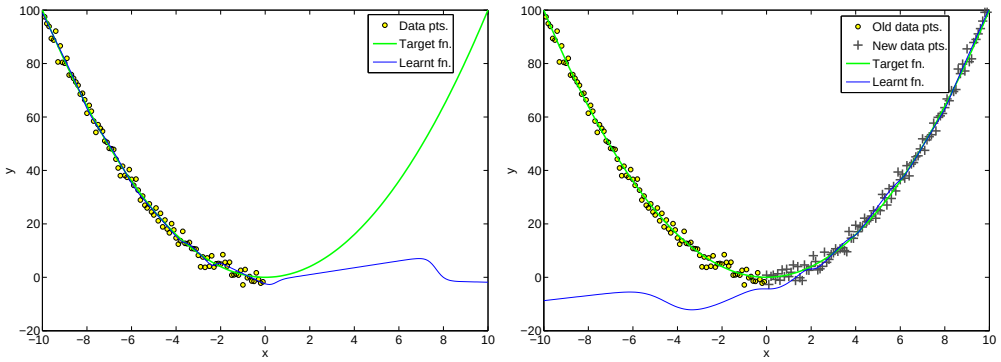


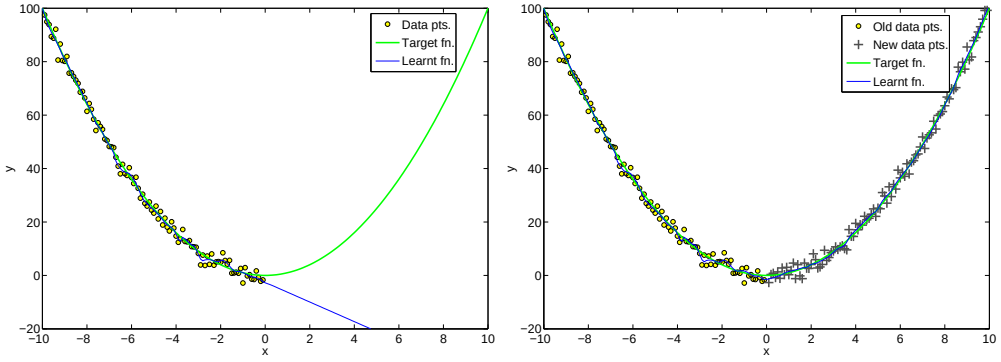
Figure 1.5: Illustration of function approximation using a mixture of regression experts

In a mixture model, the region of locality is learnt by splitting the input region amongst the local models that make up the mixture. The gating function assigns a probability $P(z_i = j|\mathbf{x}_i)$ of a data belonging to the j^{th} expert and since $\sum_j P(z_i = j|\mathbf{x}_i) = 1$, there is a fraction of contribution by each expert in explaining the data. Reassigning the responsibility of a single model during training, thus affects the responsibilities of the other local models of the mixture. This again leads to the problem of negative interference amongst the local models of the mixture. The phenomenon of negative interference in a mixture model has been illustrated in Fig. 1.6(a) and Fig. 1.6(b) where data from different regions of input space are used to train the mixture model. In the first phase, data lying on the negative half of the input space is used to train the mixture model. The function approximation by the trained model after convergence of the EM algorithm is shown in Fig. 1.6(a). In the second phase, data from the positive half of the input space is used to retrain the mixture model and the resultant function approximation after the retraining is shown in Fig. 1.6(b). During these training epochs the centers of the models of the mixture model are kept constant for simplicity and ease of visualization. As can be seen from the results, the global optimization of the local models of the mixture leads to negative interference which manifests as a suboptimal fit displayed in case (b) in the input region that had previously been learnt optimally in case (a). This is primarily due to the fact that, as the input distribution changes, the responsibilities of the local models lying in the positive half change during the train-



(a) Initial presentation of data pts. to mixture model

(b) Adaptation of mixture model to modified input distribution



(c) Initial presentation of data pts. to LWPR

(d) Adaptation of LWPR to modified input distribution

Figure 1.6: Illustrative comparison of negative interference

ing which is then propagated to the models lying in the other half of the input space although these models are not directly affected by the new data. The same example is learned using LWPR which uses local experts that have independent learning routines and the results are illustrated in Fig. 1.6(c) and 1.6(d). These figures clearly illustrate the lack of negative interference when local models with uncorrelated learning routines are used. We can hence conclude that in order to avoid negative interference it is not just sufficient for the learning algorithms to be spatially localised, but is also necessary for them to have independent training routines. As a solution to this problem we develop a probabilistic model of independent ensemble learning in Chapter 2 that motivates a probabilistic formulation of independent learning through the use of a Product of Experts paradigm.

1.4 Aims and outline of the thesis

From the discussion in this chapter, we can summarize the aim of the thesis as - develop a spatially *localised online* learning algorithm with a *Bayesian* formulation, that learns local models completely *independent* of each other, adapts the local model complexity in a data driven fashion and has an *efficient* training algorithm with minimal open parameters.

Chapter 2 provides a motivation for a principled probabilistic framework for independent ensemble learning by modeling the global regression model as a Product of Experts. The probabilistic formulation allows us to perform Bayesian inference of the parameters whereas the independent training of the ensemble experts allows for dynamic adaptation of complexity of the regression model during incremental learning.

Chapter 3 provides a hierarchical probabilistic model for each expert of the ensemble termed as a Randomly Varying Coefficient model. This results in a localised regression paradigm that makes the function approximation robust against negative interference.

Chapter 4 provides the inference rules for the parameters of the local regression models using a Variational Bayesian EM. This overcomes the problems of overfitting and allows for principled model selection thus making the inference procedure fairly insensitive to parameter initializations.

Chapter 5 provides the online updates for the parameters of the regression model by readapting the Bayesian inference procedure derived in Chapter 4.

Chapter 6 extends the localised regression to the problem of classification by using a logistic regression formulation of the Randomly Varying Coefficient model.

Chapter 7 summarises the key contributions of the thesis along with possible directions for extending the work in this thesis.

Chapter 2

Regression using Product of Experts

We saw in the earlier chapter that it is essential to model the target function as a set of local linear approximations and have independent learning rules for these local models to avoid negative interference and simplify model construction. In this chapter we attempt to provide a justification of the independent training of local models by modeling the joint likelihood of all the experts as an *unnormalized product* of individual expert probabilities. This chapter provides the basic framework for combining the probabilistic predictions of individual local models in line with the ideas of *committee machines* and *ensemble learning*.

2.1 Committee machines

There has been extensive research into committee machines to make use of a distributed architecture of learning such that individual models specialize differentially (Brown et al., 2005a; Dietterich, 2002; Freund & Schapire, 1996). The idea of committee machines originated when it was found that a combination of multiple learners that were trained using the same data provided more robust predictions than a single model trained on the same data. One of the noteworthy case of ensemble learning has been the method of *boosting* (Schapire, 1999). Using this procedure a set of *weak* learners are trained on the training data with the data weighted differently for the different learners and individual predictions from these learners are then combined together. This procedure has been found to produce powerful learners although the base learners are themselves weak.

There can be two variants of committee learning - one where the number of learners are fixed and the learning is competitive with the learners trying to share amongst them-

selves the responsibility of explaining the data. In the other case the learners are trained independent of each other and the predictions of these learners are combined together. Mixture of experts (Xu et al., 1995) fall under the former category. For reasons explained in Chapter 1, we are interested in the latter category of learning and we would implicitly refer to the latter case when using the term committee machines. Although there has been a lot of research into independent committee machines (Kuncheva et al., 2000; Demirekler & Altinçay, 2002; Hashem & Schmeiser, 1995), the theoretical justification of independent training has not been forthcoming. Especially it has been rather difficult to come up with a probabilistic formulation of an independent committee machine. In this chapter we use a product of experts (POE)(Hinton, 1999) as the probabilistic equivalent of committee machines and illustrate an approximation of POE to make the learning of the components independent.

2.2 Product of regression experts

Let us formulate the conditional distribution of the response variable as a product of local distributions. Specifically we model the distribution of the response as a product of local Gaussian distribution with a parametric mean function centered at $\mathbf{x}_c$ with a heteroscedastic[1] noise component given by :

$$y|\mathbf{x},\mathbf{x}_c \sim \mathcal{N}(f(\mathbf{x}), 1/\phi(\mathbf{x}-\mathbf{x}_c)) \tag{2.1}$$

For a given input $\mathbf{x}$, the local model centered at $\mathbf{x}_c$ predicts the output as $f(\mathbf{x})$ with a confidence proportional to $\phi(\mathbf{x}-\mathbf{x}_c)$ which has a form similar to the weighting function in eq. (1.2). Local models centered at different locations in the input space can now be combined by taking the normalized product of the probabilities. The combined conditional probability is given by :

$$y|\mathbf{x} \sim \frac{\prod_{j=1}^{M} \mathcal{N}(f_j, 1/\phi_j)}{Z} \tag{2.2}$$

where $f(\mathbf{x})$ has been abbreviated to f, $\phi(\mathbf{x}-\mathbf{x}_c)$ has been abbreviated to ϕ, j is the index over the local models of the ensemble and Z is the normalization constant given by :

$$Z = \int \prod_j \mathcal{N}(y; f_j, 1/\phi_j) dy$$

[1] input dependent noise variance

The formulation in eq. (2.2) is called the product of experts. Here, individual experts are defined by eq. (2.1) and their product combination by eq. (2.2) which is another Normal distribution given by :

$$y|\mathbf{x} \sim \mathcal{N}(\frac{\sum_j \phi_j f_j}{\sum_j \phi_j}, 1/\sum_j \phi_j) \tag{2.3}$$

We find that the mean prediction is given by the sum of predictions of individual local models weighted by the confidence of each model about its prediction. This is similar to linear combination rule of regressors found in committee machines (Kittler et al., 1998). This connects the probabilistic formulation of product of experts with the more conventional treatment of committee machines. Parameters of the POE can now be learnt by maximizing the likelihood defined in eq. (2.2) but this introduces dependency between the local models due to the normalization term. We can eliminate the dependency by ignoring the normalization term; resulting in maximization of the unnormalized likelihood of a POE while still using the normalized version for computing the prediction. This has sparked criticism of independent ensemble learning claiming that the models used for training and prediction are different. Although the criticism is well founded, due to reasons of computational efficiency and the need for dynamic adaptation of model complexity during online learning, we retain the independent learning scenario in this thesis.

The idea of independent learning of the components of an ensemble have appeared in different guises in slightly disparate fields of machine learning. In the next few sections we try to unify these ideas using the mathematical framework afforded by POE serving as a common ground to bridge these different definitions of independent learning. Here, the attempt is not to justify independent learning but to unify the different views of independent learning.

2.2.1 Diversity formulation of POE

One of the early attempts in explaining regression ensembles has been (Krogh & Vedelsby, 1995) who showed that the squared loss for an ensemble learner is less than the sum of the squared losses of the individual models of the ensemble. This can be derived by splitting the squared loss of an ensemble into a sum of squared losses of individual models and a *diversity* (Kuncheva & Whitaker, 2003) term. The same derivation holds true for the log-likelihood resulting from the product of experts (POE) combination of Normal distributed regression components. Writing down the

log of the likelihood given by eq. (2.2) we get :

$$\mathcal{L} = \sum_j -\frac{\phi_j (y - f_j)^2}{2} + \frac{1}{2}\ln(\phi_j) - \ln(Z) \tag{2.4}$$

When Z is expanded, the equation can be rewritten as -

$$\begin{aligned}
\mathcal{L} &= \sum_j -\frac{\phi_j (y - f_j)^2}{2} + \frac{1}{2}\sum_j \phi_j f_j^2 - \frac{1}{2}\frac{(\sum_j \phi_j f_j)^2}{\sum_j \phi_j} + \frac{1}{2}\ln\sum_j \phi_j \\
&= \sum_j -\frac{\phi_j (y - f_j)^2}{2} + \frac{1}{2}\sum_j \phi_j (f_j - \frac{\sum_j \phi_j f_j}{\sum_j \phi_j})^2 + \frac{1}{2}\ln\sum_j \phi_j \\
&= \sum_j -\frac{\phi_j (y - f_j)^2}{2} + \frac{1}{2}\sum_j \phi_j (f_j - f_{ens})^2 + \frac{1}{2}\ln\sum_j \phi_j
\end{aligned} \tag{2.5}$$

where $f_{ens} = \frac{\sum_j \phi_j f_j}{\sum_j \phi_j}$ is the prediction of the ensemble model. This equation shows the correspondence of the likelihood of a POE model to the *ambiguity* decomposition derived by (Krogh & Vedelsby, 1995) and explained in depth in (Brown et al., 2005b). The second term of equation eq. (2.5) is referred to as the ambiguity term and essentially measures the diversity of individual models of the ensemble. There have been many methods of ensemble learning that strive to achieve higher generalization ability by trying to increase this diversity (Brown et al., 2005a). In (Brown et al., 2005b), a generalized version of the ambiguity based loss was provided as :

$$\mathcal{L} \propto \sum_j -\frac{\phi_j (y - f_j)^2}{2} + \lambda \sum_j \phi_j (f_j - f_{ens})^2$$

where, λ modulates the contribution of the diversity of the ensemble to the loss function. Setting λ to zero, then will correspond to independent learning.

To understand the effect of an independent learning assumption, consider a localised regression where the confidence $\phi(\mathbf{x} - \mathbf{x}_c)$ is a symmetric decaying function about the center $\mathbf{x}_c$, like a Gaussian. This would mean that at the center of a local model k the ensemble prediction f_{ens} is dominated by the prediction of the k^{th} local model with the weight of the local model being higher than others and as $\mathbf{x} \to \mathbf{x}_c$, $f_{ens} \to f_k$ and $\forall_{j \neq k} \phi_j \to 0$. This results in the diversity term itself going to zero. For x away from the centers and lying in the region of overlap of different local models, the diversity term is finite and the error due to the unnormalized likelihood increases for these points. However, it is quite difficult to derive a generalized theoretical bound for the error caused due to the independence assumption.

2.2.2 Independent learning using Complementary prior

Independent learning of ensembles can also be motivated as a form of prior that can be used to eliminate the inter-dependence introduced due to the normalization factor of the likelihood. Unlike in the previous section, we are now interested in making proper Bayesian inference over the parameters of the model. To this end, we have to find the posterior probability of the local function estimate f_j given a certain prior probability over it. We show in this section that by carefully choosing the form of the prior probability over the function estimates it is possible to get an inference process that is independent.

We use a *complementary* prior as formulated by Hinton et al. in (Hinton et al., 2006), where it was used to overcome the *explaining away* phenomenon in directed graphical models. The prior is designed so that the parameters of the prior distribution are tied together in a fashion that is complementary to that of the likelihood and hence cancels it out when computing the posterior. A similar prior was also discussed in (Murray & Ghahramani, 2004) in the context of inference in a Boltzmann machine though the authors of the work dismissed such a prior citing its dependence on the data.

In this thesis, we try to derive a complementary prior for the POE regression model at hand. To start with, it is not necessary that such a complementary prior exist at all (refer to (Hinton et al., 2006) for conditions under which such a prior exists), but our model being a simple Gaussian it is possible to derive such a prior. To derive the prior we start off by transforming the normalization constant into a distribution of the parameters. Given a conditional POE model :

$$P(y|\theta_1 \dots \theta_j \dots \theta_M) = \frac{\prod_j P(y|\theta_j)}{\int \prod_j P(y|\theta_j) dy} \tag{2.6}$$

we can come up with a joint prior for $\theta_1 \dots \theta_j \dots \theta_M$ as -

$$P(\theta_1 \dots \theta_j \dots \theta_M) \propto \int \prod_j P(y|\theta_j) dy \prod_j Q(\theta_j) \tag{2.7}$$

where $Q(\theta_j)$ is an arbitrary probability distribution over θ_j. Using Bayesian rule, we can combine the likelihood defined by eq. (2.6) and the prior as defined by eq. (2.7) to get the posterior over the parameters as :

$$P(\theta_1 \dots \theta_j \dots \theta_M|y) \propto \prod_j P(y|\theta_j) Q(\theta_j) \tag{2.8}$$

$$\Rightarrow P(\theta_j|y) \propto P(y|\theta_j) Q(\theta_j) \tag{2.9}$$

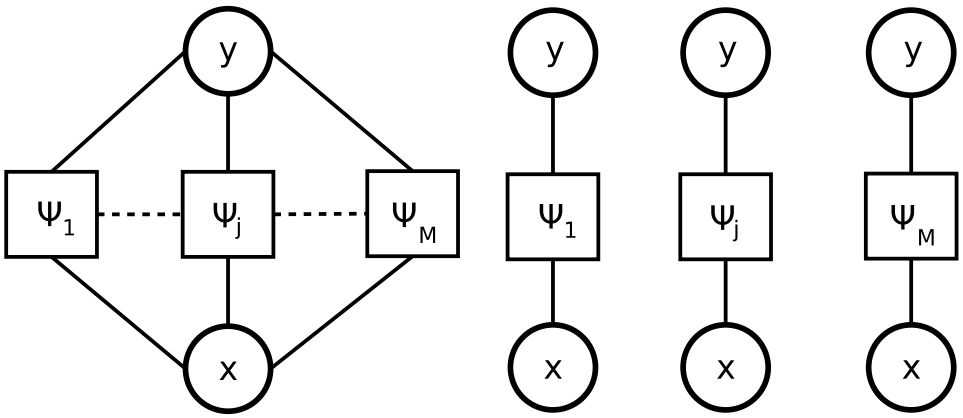


Figure 2.1: (a) Undirected graph model for POE based regression, y and x are the output and input variables for regression and M_j are the local models of the ensemble (b) node-split approximation for independent learning

where in eq. (2.8), the inference is rendered independent when the normalization term gets cancelled by the complementary part of the prior probability.

The complementary prior for the fits of the local models of the POE regression can be derived by assuming that the weight function ϕ_j is fixed. The prior over the fits can then be written down as :

$$P(f_1 \dots f_j \dots f_M) \propto \int \prod_j P(y|f_j) dy \prod_j Q(f_j|\alpha)$$

We can see from the equation that the complementary prior is equivalent to assuming prior correlations between the parameters of the ensemble models. The correlation term in the complementary prior probability is designed to cancel out the correlations amongst the parameters introduced by the likelihood term, making the posterior probabilities over the parameters independent.

2.2.3 POE regression as a conditional random field

Finally we present POE as a conditional random field and demonstrate the effect of independent learning under such a setting. A *conditional random field* (Lafferty et al., 2001) is a representation of a conditional distribution as a Markov random field. A conditional random field is any conditional distribution that can be expressed as a normalized product of functions(usually termed *factors*) :

$$P(y|x) = \frac{\prod_j \Psi_j(y,x)}{Z}$$

where ψ are the local factors and Z the normalization constant. One can immediately note that the POE regression model discussed in the earlier section is a conditional random field and can be represented as a factor graph (Kschischang et al., 2001) as shown in Fig. 2.1(a). In a factor graph as in Fig. 2.1(a), random variables are (denoted as circles) connected to all the factors (denoted as squares) in which it appears. The paradigm of independent learning of ensemble model can now be viewed as a node-splitting approximation similar to (Sutton & McCallum, 2005) where the nodes are split into duplicate nodes and inference is performed on independent disconnected components of the factor graph. The resulting model for POE regression is shown in Fig. 2.1(b). In (Sutton & McCallum, 2005) independent learning has been justified as maximizing a lower bound on the loss function defined over the entire ensemble, though their derivation of the bound is restricted to a certain family of parameterization. The same derivation can be applied to the POE model of regression if the weight function $\phi(x - x_c) < 1$. To derive the relation between the dependent likelihood and independent likelihood, we write down the likelihoods for the cases represented by the graphical models in Fig. 2.1. The dependent log likelihood is given by :

$$\mathcal{L}_{dep} = \sum_j -\frac{\phi_j (y - f_j)^2}{2} + \frac{1}{2}\sum_j \phi_j (f_j - f_{ens})^2 + \frac{1}{2}\ln\sum_j \phi_j$$

where ϕ_j is a shorthand notation for the weighting function of the jth local model - $\phi_j \equiv \phi(x - x_j)$ and the independent likelihood is just the sum of the likelihoods of the individual disconnected factor graphs of Fig. 2.1(b) and is given by :

$$\mathcal{L}_{ind} = \sum_j -\frac{\phi_j (y - f_j)^2}{2} + \frac{1}{2}\sum_j \ln \phi_j$$

Taking the difference of the likelihoods we get :

$$\begin{aligned}
\mathcal{L}_{dep} - \mathcal{L}_{ind} &= \sum_j \phi_j (f_j - f_{ens})^2 + \ln\sum_j \phi_j - \sum_j \ln \phi_j \\
&> \ln\sum_j \phi_j - \sum_j \ln \phi_j \quad \text{since, } \phi > 0 \Rightarrow \sum_j \phi_j (f_j - f_{ens})^2 > 0 \\
&> 0 \quad \text{using Jensen's inequality when } \forall_j\ \phi_j < 1
\end{aligned} \tag{2.10}$$

The above equations prove that the likelihood for an independent regression POE model is a lower bound of the likelihood corresponding to the dependent model. By maximizing the independent likelihood we are in effect maximizing the lower bound of the actual objective function. The condition holds only when the weight function ϕ is less than one. This property is satisfied by weighting functions that have an appropriate

kernel function like an exponential function of the form $exp(-(\mathbf{x}-\mathbf{x}_c)^T(\mathbf{x}-\mathbf{x}_c))$. For an inverse polynomial function like the one used in this thesis ϕ can be restricted to a maximum of one by scaling the inputs appropriately.

2.3 Discussion

In this chapter we have reviewed the definition of independently trained ensemble models in recent research. Despite the difficulty in deriving a strong theoretical justification of independent learning, empirical evidence presented in (Sutton & McCallum, 2005) shows that the independent likelihood is sufficient to obtain good parameter estimates for the model. When coupled with the ease of training in a constructive and incremental learning setup, this makes independent ensemble models an attractive option for efficient online learning. This motivates the use of independent regression ensemble in this research.

Chapter 3

Randomly Varying Coefficient model

In the previous chapter we looked at modeling regression as a product of locally linear experts and studied the properties of such a model. In this chapter we concentrate on the local models constituting the product of experts and formulate a probabilistic model for the local linear regression.

Modeling spatially localized linear models using a probabilistic framework involves deriving a formulation that allows to model the *fit*, in our case a linear fit, and the *bandwidth* at a particular location in the input space. Each of these local models can then be combined to provide a prediction for a novel data. Additionally, in order for the local models to be independent, each of them should be capable of modeling the entire data by learning the correct bandwidth that partitions the data into two parts – one which corresponds to the linear region of interest and the other which does not. In this thesis, we accomplish this by formulating a probabilistic model called Randomly Varying Coefficient(RVC) model which builds upon the idea of a random coefficient model (Longford, 1993).

For a locally linear region centered around $\mathbf{x}_c$ a generative model for the data points can be written as:

$$y_i = \beta_i^T \mathbf{x}_i + \varepsilon \tag{3.1}$$

where $\mathbf{x}_i \equiv [(\mathbf{x}_i' - \mathbf{x}_c)^T, 1]^T$ represents the center subtracted, bias augmented input vector, $\beta_i \equiv [\beta_i^{(1)} \dots \beta_i^{(d+1)}]^T$ represents the corresponding regression coefficient and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is the Gaussian mean zero noise with a standard deviation σ. The data is assumed to have been generated in an IID fashion. Crucially, we allow the regression coefficient to be a random variable with a prior distribution given by:

$$\beta_i \sim \mathcal{N}(\hat{\beta}, \mathbf{C}_i) \tag{3.2}$$

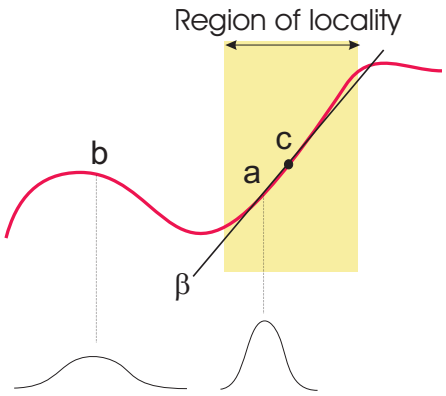


Figure 3.1: Variation of prior with the location of the input

where we have assumed that each β_i is generated from a Gaussian centered around $\hat{\beta}$ with the confidence being represented by the covariance $\mathbf{C}_i$. The covariance itself is defined to be proportional to the distance of $\mathbf{x}'_i$ from the center. This has the effect that for points that lie close to the center, the distribution of β_i is peaked around $\hat{\beta}$ resulting in a linear region around the center. This has been illustrated schematically in Fig. 3.1 where point c is the center of the local model: for a point a that lies close to c we assign a prior that is fairly tight around the mean whereas for a point b that lies away from c the prior is much broader. One can consider various distance functions to index the variation of the covariance matrix $\mathbf{C}$. Here, we restrict ourselves to a diagonal version, each diagonal element varying quadratically with $\mathbf{x}$ as:

$$\mathbf{C}_i(j,j) = ((\mathbf{x}'_i - \mathbf{x}_c)^T(\mathbf{x}'_i - \mathbf{x}_c) + 1)/h_j^2 = \mathbf{x}_i^T\mathbf{x}_i/h_j^2 \tag{3.3}$$

where h_j is the *bandwidth* parameter of the kernel defining the extent of the locality along the j-th dimension and $\mathbf{x}_i$ is the center subtracted input variable. Larger values of h_j implies a larger extent of the local region and vice versa.

3.1 Varying coefficient model

Randomly varying coefficient model is based upon a varying coefficient model where the regression coefficient of a linear regressor varies with the data and this approach has found wide application in statistics as a specialization of multilevel models (Gelman & Hill, 2007). In a varying coefficient model the regression coefficient is varied to model the differences between different classes of data. For instance, in (Price et al., 1996) the radon concentration in different households is analyzed using a varying intercept,

varying slope model. Different intercepts and slopes are used to model the variance in Radon levels between counties where the counties are the different classes. This is similar to a classical mixture model (Xu et al., 1995) where different parameters are assigned to different clusters defined over the data. This approach is useful when the classes in the data are well defined and it is possible to define a function that discriminates between the different classes. In our case, where we need to carve out a region of linearity for an independent local model there are no competing models that can stand for distinct classes. To appreciate the difficulty in formulating the problem, we could try to model a regression using two classes - one class corresponds to a model that is responsible for the linear region and the other class models the rest of the data. Though it is easy to come up with a probabilistic model that corresponds to the linear region, it would be difficult to come up with a model that can model its complement. This is mainly due to the fact that we cannot assign any prior belief on the model in the "non" linear region. In this research we use a single class and then assign a prior probability of a data point being generated from that class based on the location of the data point in the input space. Here, the class is represented by the Gaussian distribution over the mean parameter $\hat{\beta}$ and the individual data from this class is the hidden variable β. Hence, unlike the conventional multilevel model, there are as many hidden variables[1] β as there is data. One could imagine using a box car function for modeling the covariance in eq. (3.3) such that the variance is low and constant inside the linear region and high in the region outside. This would be a special case where the input space is differentiated distinctly into a region of linearity and its complement. The quadratic function of eq. (3.3) on the other hand corresponds to a soft partition of the input space.

3.1.1 Pooling in the hierarchical model

One of the reasons why multilevel models have been used is the way the parameter is estimated as a combination of the variables at the levels below and above it on the probabilistic hierarchy termed as *pooling*. RVC model utilizes the pooling phenomenon to estimate the hidden variable β. Consider the two-level hierarchy defined by eq. (3.1) and eq. (3.2), if we assume that h is known and hence $\mathbf{C}$, then the posterior of the hidden variable β is given by :

$$\tilde{\beta}_i \sim \mathcal{N}(\nu_i, \mathbf{G}_i) \tag{3.4}$$

[1] Although the parameters of the model are also hidden, we reserve the term *hidden variable* to refer to the variable whose cardinality increases with the data

where,

$$\begin{aligned}
\mathbf{G}_i &= (\mathbf{x}_i\mathbf{x}_i^T/\sigma^2 + \mathbf{C}_i^{-1})^{-1} \\
&= \mathbf{C}_i - \frac{\mathbf{C}_i\mathbf{x}_i\mathbf{x}_i^T\mathbf{C}_i}{\sigma^2 + \mathbf{x}_i^T\mathbf{C}_i\mathbf{x}_i} \quad \text{by Sherman-Morrison Woodbury theorem} \\
\nu_i &= \mathbf{G}_i(y_i\mathbf{x}_i/\sigma^2 + \mathbf{C}_i^{-1}\hat{\beta}) \\
&= \frac{\mathbf{C}_i\mathbf{x}_i}{(\sigma^2 + \mathbf{x}_i^T\mathbf{C}_i\mathbf{x}_i)}\left(y_i - \mathbf{x}_i^T\hat{\beta}\right) + \hat{\beta} \qquad (3.5)
\end{aligned}$$

We can see the effect of pooling more clearly, if we were to premultiply eq. (3.5) by $\mathbf{x}_i^T$:

$$\begin{aligned}
\mathbf{x}_i^T\nu_i &= \frac{\mathbf{x}_i^T\mathbf{C}_i\mathbf{x}_i}{(\sigma^2 + \mathbf{x}_i^T\mathbf{C}_i\mathbf{x}_i)}y_i + (1 - \frac{\mathbf{x}_i^T\mathbf{C}_i\mathbf{x}_i}{(\sigma^2 + \mathbf{x}_i^T\mathbf{C}_i\mathbf{x}_i)})\mathbf{x}_i^T\hat{\beta} \\
&= \omega_i y_i + (1 - \omega_i)\mathbf{x}_i^T\hat{\beta}
\end{aligned}$$

where $\omega_i = \frac{\mathbf{x}_i^T\mathbf{C}_i\mathbf{x}_i}{(\sigma^2 + \mathbf{x}_i^T\mathbf{C}_i\mathbf{x}_i)}$ is called the *pooling factor*(Gelman & Pardoe, 2006). As $\omega \to 1$ the posterior estimate ν_i tends to be pooled towards the data and as $\omega \to 0$ the estimate is closer to $\hat{\beta}$. The pooling factor ω tends to 0 when the term $\mathbf{x}_i^T\mathbf{C}_i\mathbf{x}_i$ is negligible compared to the noise term σ, which means that the prior dominates over the likelihood term and the estimate is close to the prior term (in this case $\hat{\beta}$) and vice versa. The pooling factor is plotted in Fig. 3.2 for a sample uniform grid input distribution centered around the origin. Here $\mathbf{C}$ is given by eq. (3.3), parameters h and σ were chosen arbitrarily and the local model is centered around the origin. From the plot, it is obvious that the pooling factor approaches 0 near the center of the model and approaches 1 away from the center as is expected. Also shown in the plot are the pooling factors for different values of the bandwidth h. For smaller bandwidths the curve is sharp at the bottom and is more flat for higher values of the bandwidths. This implies that for higher bandwidths the extent of data around the center of the model pooled towards the mean regression coefficient is higher and in turn leads to an increased expanse of linearity.

3.2 Locally weighted linear model

A Randomly Varying Coefficient model can also be understood as a local linear regressor. The local regression equivalent to RVC can be obtained by marginalizing out the hidden variables β_i of the local model to obtain :

$$\begin{aligned}
P(y_i|\hat{\beta}, \sigma, h_1 \ldots h_{d+1}) &= \int P(y_i|\beta_i^T\mathbf{x}_i, \sigma^2)P(\beta_i|\hat{\beta}, \mathbf{C}_i)d\beta_i \\
&\Rightarrow y_i \sim \mathcal{N}(\hat{\beta}^T\mathbf{x}_i, \mathbf{x}_i^T\mathbf{C}_i\mathbf{x}_i + \sigma^2) \qquad (3.6)
\end{aligned}$$

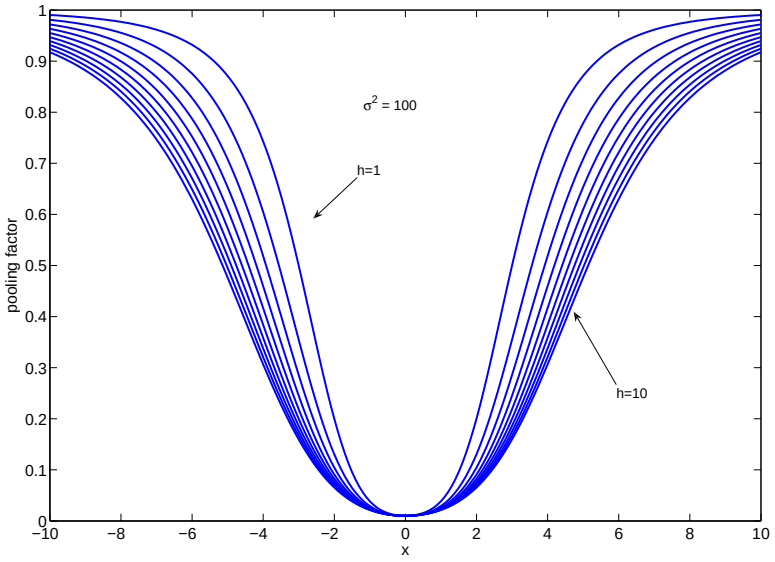


Figure 3.2: The pooling factor as a function of the input distribution

It is interesting to note that the form of likelihood in eq. (3.6) corresponds to a linear regression with heteroscedastic noise (Gelman et al., 2003). Thus the Randomly Varying Coefficient formulation strives to model the data points as being generated from a linear function with a noise process that increases monotonically with the distance from the center of the local model. This brings the model in line with the kind of heteroscedastic model we had discussed in the context of product of experts regression in Chapter 2.

Assuming IID data, the log likelihood for the entire data is given by the sum of the log likelihood of individual data points defined in eq. (3.6), and can be written as :

$$\mathcal{L} = \sum_i -\frac{1}{2}\ln(\mathbf{x}_i^T \mathbf{C}_i \mathbf{x}_i + \sigma^2) - \frac{1}{2}\frac{(y_i - \hat{\beta}^T \mathbf{x}_i)^2}{(\mathbf{x}_i^T \mathbf{C}_i \mathbf{x}_i + \sigma^2)} \tag{3.7}$$

Eq. (3.7) can be rewritten in a more generic form by replacing the variance part $\mathbf{x}_i^T \mathbf{C}_i \mathbf{x}_i + \sigma^2$ by a weighting function $\frac{1}{\phi(x_i,h)}$ to yield :

$$\mathcal{L} = \sum_i \frac{1}{2}\ln\phi(x_i,h) - \frac{1}{2}\phi(x_i,h)(y_i - \hat{\beta}^T \mathbf{x}_i)^2 \tag{3.8}$$

The log likelihood $\mathcal{L}$ can be seen to be made up of two terms - a weighted squared error term that represents the bias of the fit and the normalization term that corresponds to the variance. The weighting function contributes to both these terms with the bandwidth parameter h of the weighting function modulating the bias and variance of the fit. The

optimal bandwidth would hence be a trade-off between the bias and variance and would typically depend on the functional form of the objective function being optimized.

3.2.1 The weight function for the local linear regression

As we saw in eq. (3.8), the marginal likelihood can be understood as a weighted least square regression with appropriate regularization. The nature of the weighting is governed by the weighting function that is used to provide locality.

The weighting function in our model is given by $\phi(x_i, h) = 1/(\mathbf{x}_i^T C_i \mathbf{x}_i + \sigma^2)$. This weighting function corresponds to an inverse of a quartic polynomial in x. This type of weight function has not been used previously in local least squares regression or kernel regression. Researchers usually prefer to use weight functions like Gaussian, Epanechnikov or tricubic kernels (Härdle, 1994; Loader, 1999a). However it has been noted (Härdle, 1994; Atkeson et al., 1997) that given an optimal bandwidth for a kernel the fit is not sensitive to the shape of the kernel. Hence, in this work we have chosen an inverse polynomial keeping an eye on the ease of inference afforded by this form of kernel.

In this section we had formulated RVC as a local linear regression with parameters for the fit and bandwidth. A straightforward approach to estimate the *fit* ($\hat{\beta}$) and the *bandwidth* (h) parameters of RVC would be to optimize the log likelihood given by eq. (3.8). However, it is observed that the maximum likelihood (ML) estimate for the fit obtained using the log likelihood suffers from substantial bias at points of high curvature along the function being modeled and requires a regularizer prior over the bandwidth parameter as explained next.

3.2.2 Bias reduction for linear fit

The log likelihood function $\mathcal{L}$ given by eq. (3.8) is a typical loss function for a locally weighted regression and it is generally observed that when a localised linear regression is used to obtain a smoothed estimate of a non linear function, substantial bias is introduced at points of high curvature. This is usually referred to as "trimming the hills and filling the valley" (Hastie & Loader, 1993) and is illustrated in Fig. 3.3. The phenomenon can be demonstrated by estimating the bias for a simple *noiseless* function defined over a *one dimensional* input. The log likelihood of a local model of RVC as

defined by eq. (3.8) can be adapted to a univariate target function as :

$$\mathcal{L} = \sum_i \frac{1}{2} \ln \phi(x_i, h) - \frac{1}{2} \phi(x_i, h)(f(x_i) - m(x_i - x_c) - c)^2$$

where $f(x)$ is the function to be approximated, x_c the center of the local model, m the slope and c the intercept of the univariate regression. To estimate the value of the bias introduced by an ML estimate of the parameters, the slope and intercept of the weighted regression needs to be computed by differentiating $\mathcal{L}$ with respect to the parameters and equating to zero :

$$\frac{\partial \mathcal{L}}{\partial m} = \sum_i \phi(x_i, h)(x_i - x_c)(f(x_i) - m(x_i - x_c) - c) = 0 \tag{3.9}$$

$$\frac{\partial \mathcal{L}}{\partial c} = \sum_i \phi(x_i, h)(f(x_i) - m(x_i - x_c) - c) = 0 \tag{3.10}$$

Solving the above simultaneous equations in m and c we obtain :

$$\begin{aligned}(f(x_c) - c) &= f''(x_c) \frac{\sum_i \phi(x_i, h)(x_i - x_c) \sum_i \phi(x_i, h)(x_i - x_c)^3 - (\sum_i \phi(x_i, h)(x_i - x_c)^2)^2}{(\sum_i \phi(x_i, h)(x_i - x_c))^2 - \sum_i \phi(x_i, h) \sum_i \phi(x_i, h)(x_i - x_c)^2} \\ &= w f''(x_c) \end{aligned} \tag{3.11}$$

where $f(x_i)$ has been replaced by its Taylor expansion about x_c upto the second degree. Eq. (3.11) clearly illustrates the bias $(f(x_c) - c)$ as a function of the curvature represented here by the second order differential $f''(x_c)$.

Different methods have been proposed in the literature to reduce this bias -

- Using a higher degree polynomial fit can overcome the effect of higher degrees of the function. In the above derivation we have effectively shown that a linear fit can overcome bias effects of first degree, similarly a quadratic fit can overcome effects of second degree and so on. This has been proved for any generic local polynomial fit in (Hastie & Loader, 1993), (Fan & Gijbels, 1995).

- As illustrated in (Choi & Hall, 1998) the bias at a point of high curvature can also be reduced by using local models in neighboring regions placed in such a way as to cancel out the effects of the bias.

- If we assume that the fit is sufficiently local so that $\phi(x, h)$ has a fast decay, then the data distribution around the center of the local region can be assumed to be fairly symmetric. In addition when $\phi(x, h)$ is symmetric around x_c and always

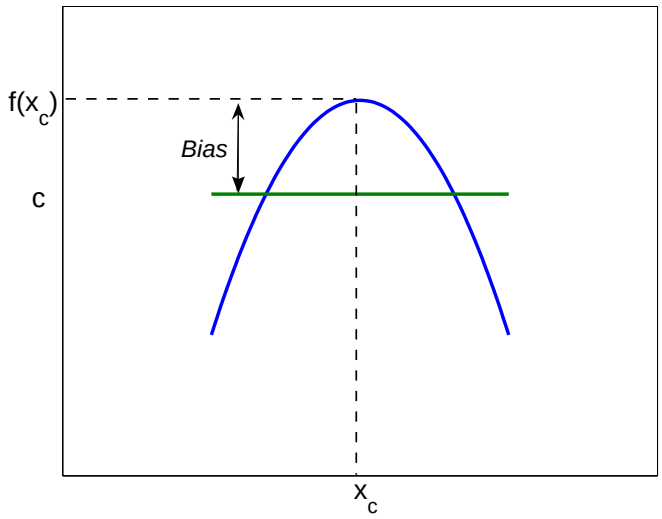


Figure 3.3: Bias for a local linear regression

nonnegative, the summation of terms of odd degree in eq. (3.11) reduces to zero. Hence, w in eq. (3.11) can be approximated as :

$$w \approx \frac{\sum_i \phi(x_i,h)(x_i-x_c)^2}{\sum_i \phi(x_i,h)} \tag{3.12}$$

which means that in order to reduce the bias one can favor a weighting function $\phi(x_i,h)$ with a small bandwidth value such that terms away from the center (large values of $(x_i-x_c)^2$) receive a significantly low weight leading to smaller values for w.

All the above methods tend to decrease the bias at the expense of an increased variance. In this work we use the last method wherein small bandwidths for local models are encouraged by using a Gamma [2] *regularizer* prior over the bandwidth parameters given by :

$$h_j^2 \sim Gamma(a_j,b_j) \tag{3.13}$$

The parameter h being a scale parameter of the Normal distribution over the hidden variables β_i, the Gamma distribution over h would be a conjugate to the Normal distribution and will serve to simplify inference procedure. We shall further assign non-informative Normal prior $\mathcal{N}(\mu,\mathbf{S})$ for the parameter $\hat{\beta}$ and a noninformative inverse Gamma[3] prior with hyperparameters c and d for σ. We use values of $\mu=\mathbf{0}, \mathbf{S}=10^3\times\mathbf{I}$,

[2] $Gamma(\theta|a,b) \sim \frac{b^a}{\Gamma(a)}\theta^{a-1}e^{(-b\theta)}$

[3] $Inv-Gamma(\theta|a,b) \sim \frac{b^a}{\Gamma(a)}\theta^{-(a+1)}e^{(-b/\theta)}$

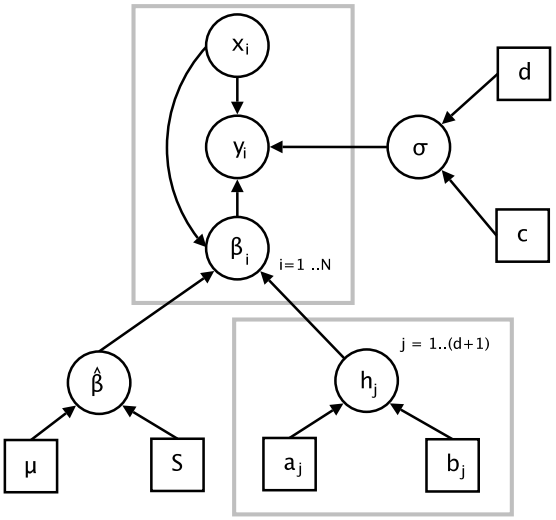


Figure 3.4: The 'local' regression model

$c = 10^{-3}$ and $d = 10^{-3}$ to make the corresponding priors non-informative. We assume a uniform prior for the regularizer hyperparameters a_j and b_j. Fig. 3.4 summarizes the resultant probabilistic model for a *single* local model.

3.3 Combining the models for prediction

In the last section, we had concerned ourselves with the building of a probabilistic model for an individual local model. In this section, we look at how these local models can be combined together to form the complete model by using the product of experts regression model discussed previously in Chapter 2. In this section, we assume that the parameters of the models have been inferred and the learnt models need to be combined during prediction.

Using the product of experts combination given by eq. (2.2), we get :

$$y \sim \frac{\prod_j \mathcal{N}(f_j, 1/\phi_j)}{Z} \tag{3.14}$$

Here, the local function f_j is modeled using a linear fit : $f_j(x^*) = \hat{\beta}_j^T(\mathbf{x}^* - \mathbf{x}_j)$ from eq. (3.8), $\mathbf{x}^*$ is the test query point and $\mathbf{x}_j$ is the center of the j^{th} local model. The product of Gaussians is just another Gaussian and hence eq. (3.14) reduces to :

$$y^*|\mathbf{x}^* \sim \mathcal{N}\left(\frac{\sum_j \phi_j(\mathbf{x}^*)\hat{\beta}_j^T(\mathbf{x}^* - \mathbf{x}_j)}{\sum_j \phi_j(\mathbf{x}^*)}, 1/\sum_j \phi_j(\mathbf{x}^*)\right) \tag{3.15}$$

We now examine the predictive process by modeling the combined prediction as a linear smoother. Maximizing the likelihood given by eq. (3.8) with respect to the fit $\hat{\beta}$ leads to a weighted regression solution. Writing down the maximum likelihood estimate of $\hat{\beta}$ we get :

$$\hat{\beta}_j = (\mathbf{X}_j^T \mathbf{W}_j \mathbf{X}_j)^{-1} \mathbf{X}_j^T \mathbf{W}_j \mathbf{y} \tag{3.16}$$

where, $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_i \dots \mathbf{x}_N]^T$, $\mathbf{W}_j$ is a diagonal matrix with the diagonal elements given by $\mathbf{W}_j(i,i) = \phi_j((\mathbf{x}_i - \mathbf{x}_j), h)$ and $\mathbf{y} = [y_1 \dots y_i \dots y_N]^T$. We can rewrite eq. (3.16) as $\hat{\beta}_j = \mathbf{S}_j \mathbf{y}$. Substituting this form in eq. (3.15) we get the mean prediction as :

$$\begin{aligned} \mathbb{E}(y^*) &= \frac{\sum_j \phi_j(\mathbf{x}^*) \hat{\beta}_j^T (\mathbf{x}^* - \mathbf{x}_j)}{\sum_j \phi_j(\mathbf{x}^*)} \\ &= \frac{\sum_j \phi_j(\mathbf{x}^*) (\mathbf{x}^* - \mathbf{x}_j)^T \mathbf{S}_j}{\sum_j \phi_j(\mathbf{x}^*)} \mathbf{y} \\ &= \mathbf{s}(x^*)^T \mathbf{y} \end{aligned} \tag{3.17}$$

where, $\mathbf{s}(x^*) = \frac{\sum_j \phi_j(\mathbf{x}^*) \mathbf{S}_j^T (\mathbf{x}^* - \mathbf{x}_j)}{\sum_j \phi_j(\mathbf{x}^*)}$. From eq. (3.17) we can conclude that the RVC model is a linear smoother (Hastie & Tibshirani., 1990; Loader, 1999a), predicting the response to a new query point as a smooth over the target points $\mathbf{y}$. The weight vector $\mathbf{s}(x^*)$ is usually called the *weight function* (Silverman, 1984) or the *weight diagram* (Loader, 1999a). It must be noted that the weight function $\mathbf{s}$ is a function of the design points and is completely independent of the response variable. For a fixed design with input data being regularly spaced within an interval, Silverman compares the weight function to a kernel and calls it an *equivalent kernel* (Silverman, 1984; Sollich & Williams, 2005). The equivalent kernel can be used to deduce the properties of the smoother as it indicates the weight received by each of the response points in accordance with its position in space. For the weight function defined in eq. (3.17), the equivalent kernel is plotted in Fig. 3.5 for three different test points with the train points being equally spaced along the input dimension.

3.3.1 Degrees of freedom

One of the ways to compare different linear smoothers is by measuring the degree of smoothness of the curve it models. The smoothness is often measured in terms of the degrees of freedom of the fit. The degrees of the freedom is also an indicator of the complexity of the learning model (in the sense of Occam's razor) and loosely corresponds to the number of parameters used by the model. The degrees of freedom

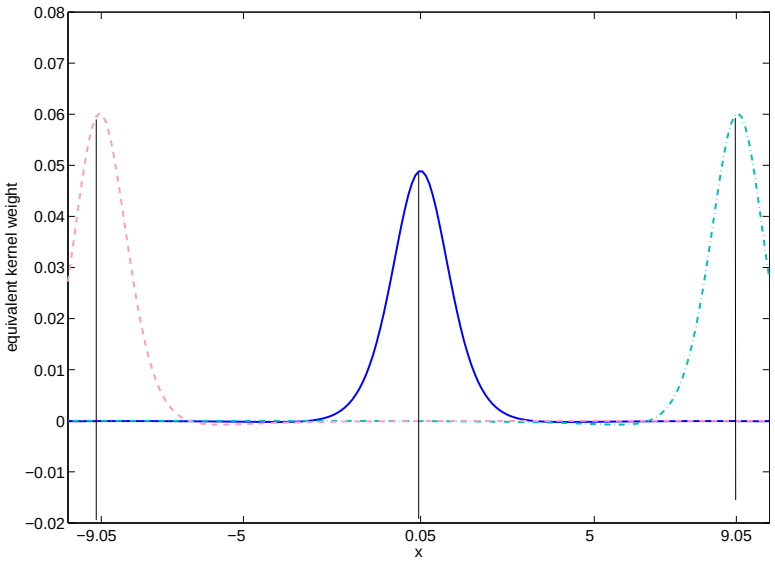


Figure 3.5: The weights produced by the equivalent kernel at different test points for a uniformly spaced train points.

of a smoother can be computed by looking at the trace of the *hat* matrix (Hastie & Loader, 1993). The *hat* matrix is defined as the matrix that maps the responses to the fits produced by the model. We can derive the hat matrix for RVC by starting with the local model. For a local model the fit is linear and hence the hat matrix $\hat{\mathbf{S}}$ is given by -

$$\begin{aligned} \hat{\mathbf{S}}_j &= \mathbf{X}_j \mathbf{S}_j \\ &= \mathbf{X}_j (\mathbf{X}_j^T \mathbf{W}_j \mathbf{X}_j)^{-1} \mathbf{X}_j^T \mathbf{W}_j \end{aligned}$$

such that $\hat{\mathbf{y}}_j = \hat{\mathbf{S}}_j \mathbf{y}$ where $\hat{\mathbf{y}}_j$ is the vector of predictions obtained from the jth local model corresponding to the vector of training responses $\mathbf{y}$. The combined estimate produced by RVC is a convex combination of the estimates of different local models :

$$\begin{aligned} \hat{\mathbf{y}} &= \sum_j \hat{\mathbf{W}}_j \hat{\mathbf{y}}_j \\ &= \sum_j \hat{\mathbf{W}}_j \hat{\mathbf{S}}_j \mathbf{y} \end{aligned} \tag{3.18}$$

where, $\hat{\mathbf{W}}_j$ is a diagonal normalized weight matrix with the diagonal elements given by $\hat{\mathbf{W}}_j(i,i) = \frac{\phi_j((x_i - x_j),h)}{\sum_j \phi_j((x_i - x_j),h)}$ which can also be expressed in terms of $\mathbf{W}_j$ as $\hat{\mathbf{W}}_j = (\sum_j \mathbf{W}_j)^{-1} \mathbf{W}_j$. From eq. (3.18), the effective weighted hat matrix would be $\hat{\mathbf{S}} = \sum_j \hat{\mathbf{W}}_j \hat{\mathbf{S}}_j$. The degree of freedom of the RVC model according to the measure developed in (Hastie & Loader, 1993) would be given by the trace of the hat matrix $\hat{\mathbf{S}}$

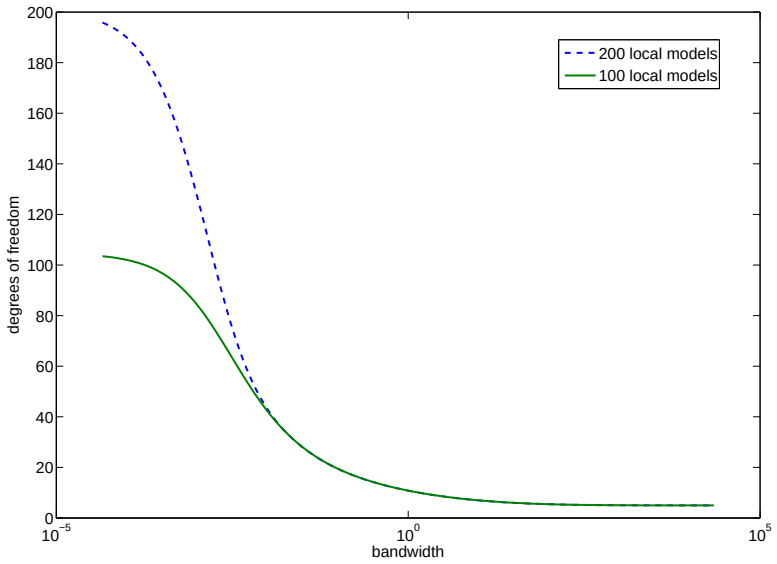


Figure 3.6: Variation of degrees of freedom with bandwidth and the number of models in RVC

given by :

$$df = trace(\hat{\mathbf{S}}) = trace(\sum_j \hat{\mathbf{W}}_j \hat{\mathbf{S}}_j) \tag{3.19}$$

It can be easily seen that the degree of freedoms of the combined system is greater than a single local model :

$$trace(\hat{\mathbf{S}}) \geq trace(\hat{\mathbf{S}}_j) \forall_{j \in 1 \ldots M} \tag{3.20}$$

For a local model having a linear parametrization for the function, $trace(\hat{\mathbf{S}}_j)$ would be equal to the number of dimensions of the input d. Therefore from eq. (3.20) we find that the combined model of RVC would have atleast as many degrees of freedom($trace(\hat{\mathbf{S}})$) as the number of dimensions of the input. As we can see from eq. (3.19), the degrees of freedom of the system is governed by the weighting provided to individual hat matrix. The weighting in turn is modulated by the individual bandwidths of the model. We can then expect that the degrees of freedom would be a function of the bandwidth. The variation of the degrees of freedom with the bandwidth of the models is illustrated in Fig. 3.6. Here, local models were centered around all the input points and were assigned the same bandwidth. The value of the bandwidth was varied and the effective degrees of freedom computed and plotted. The plot shows that as we increase the bandwidth the modeling ability of the model decreases as the degrees of freedom approaches that of a simple linear regression model and as the bandwidth decreases more complex surfaces can be modeled as it captures more vari-

ations in the given function. The other factor that influences the degrees of freedom is the number of models in the system. In Fig. 3.6 two plots have been shown to illustrate the effect of number of local models on the degrees of freedom of the system. The plot in the dashed line shows an RVC model with as many local models as data points and the plot with solid curve stands for the degrees of freedom for a case when only half of the data points are used as centers of the local models of RVC. The interesting fact to note in the figure is that as the bandwidth decreases in the top plot the degrees of freedom tends to reach the maximum which is the number of data points whereas in the bottom curve the maximum degrees of freedom exceeds the number of models in the system (in this case 100). This shows that the combined system can reach degrees of freedom exceeding the number of models. It must be noted that the illustration here has been obtained by restricting the bandwidths to be the same for all local models, this then represents a restricted subspace of the combination of the bandwidth values possible and represents a subset of the possible degrees of freedoms.

3.4 Discussion

In this chapter we had developed a probabilistic model for local linear regression and showed its links to related research literature. Furthermore we derived the complexity of the learning model for a local component of the ensemble and extended it to the entire ensemble. We also demonstrated in Section 3.3.1, that by learning the optimal bandwidth in a data driven fashion it is possible to modulate the flexibility of the model and by dynamically allocating and deallocating models in the system it is possible to enhance or limit the complexity of the system. This would be the basis for the learning algorithm of RVC developed in succeeding chapters.

Chapter 4

Learning

Learning the parameters of the Randomly Varying Coefficient model implies optimizing the joint likelihood of the variables of the model. A simple maximum likelihood estimate of the parameters would tend to produce a biased (MacKay, 1992) model which overfits the training data. An alternative would be to to learn a distribution over the parameters that gives the most probable estimate of the parameter and confidence bounds over the estimates. Use of a *non-informative* prior over the nuisance parameters results in a posterior that is insensitive to the specific prior used and thus avoids tuning of the hyperparameters. The posterior over the parameters can then be used to marginalize out the variables during prediction. This is equivalent to taking an average of the predictions of different models and results in a more robust prediction. In the RVC model, we have non-informative priors over the parameters $\hat{\beta}$ and σ (refer Fig. 3.4). For the bandwidth parameter h we have an informative prior with hyperparameters a_j and b_j. In a true Bayesian setting a non-informative prior over the hyperparameters $\mathbf{a}$ and $\mathbf{b}$ would be used to infer its posterior. However, due to the absence of any suitable conjugate prior over $\mathbf{a}$ and $\mathbf{b}$ we resort to inferring the point estimates of the hyperparameters.

Our objective is thus to learn the posterior over the parameters $\hat{\beta}$, h_j, σ and to obtain point estimates for the hyperparameters – a_j, b_j. The joint posterior is given by:

$$P(\mathbf{h},\hat{\beta},\sigma|\mathbf{y},\mathbf{a},\mathbf{b},c,d,\mu,\mathbf{S}) = \frac{P(\mathbf{y},\hat{\beta},\mathbf{h},\sigma,\mathbf{a},\mathbf{b},c,d,\mu,\mathbf{S})}{P(\mathbf{y},\mathbf{a},\mathbf{b},c,d,\mu,\mathbf{S})} \tag{4.1}$$

where we have used $\mathbf{h}$ to denote the vector $[h_1^2 \ldots h_{d+1}^2]^T$ and $\mathbf{y}$ denotes the training data $[y_1 \ldots y_N]^T$, $\mathbf{a} \equiv [a_1 \ldots a_{d+1}]^T$ and $\mathbf{b} \equiv [b_1 \ldots b_{d+1}]^T$. However, the posterior over the parameters is rendered intractable due to the difficulty in evaluating the denominator of eq. (4.1). This necessitates the use of variational Bayesian EM to evaluate the posterior

$P(\mathbf{h}, \hat{\beta}, \sigma | \mathbf{y}, \mathbf{a}, \mathbf{b}, c, d, \mu, \mathbf{S})$ and learn the regularizer hyperparameters $\mathbf{a}$ and $\mathbf{b}$.

4.1 Variational approximation for RVC

To learn the parameters of the model we can maximize the marginal log likelihood with respect to the parameters treating β_i as the hidden variables. The marginal log likelihood is given by:

$$\begin{aligned}
\mathcal{L} &= \ln P(\mathbf{y}|\mathbf{a}, \mathbf{b}, c, d, \mu, \mathbf{S}) \\
&= \ln \int P(\mathbf{y}, \beta_1 \dots \beta_N, \mathbf{h}, \hat{\beta}, \sigma^2 | \mathbf{a}, \mathbf{b}, \mu, \mathbf{S}, c, d) d\beta_1 \dots d\beta_N d\mathbf{h} d\hat{\beta} d\sigma^2 \\
&= \ln \int \left[\prod_i P(y_i|\beta_i, \sigma^2) P(\beta_i|\hat{\beta}, h_1, \dots h_{d+1}) \right. \\
&\quad \left. \prod_j P(h_j^2|a_j, b_j) P(\hat{\beta}|\mu, \mathbf{S}) P(\sigma^2|c, d) \right] d\beta_1 \dots d\beta_N dh_1 \dots dh_{d+1} d\hat{\beta} d\sigma^2 \quad (4.2)
\end{aligned}$$

Using Jensen's inequality, the objective function that lower bounds $\mathcal{L}$ for a distribution Q is given by:

$$\mathcal{F} = \int Q(\beta_1 \dots \beta_N, \mathbf{h}, \hat{\beta}, \sigma^2) \ln \frac{P(\mathbf{y}, \beta_1 \dots \beta_N, \mathbf{h}, \hat{\beta}, \sigma^2 | \mathbf{a}, \mathbf{b}, \mu, \mathbf{S}, c, d)}{Q(\beta_1 \dots \beta_N, \mathbf{h}, \hat{\beta}, \sigma^2)} d\beta_1 \dots d\beta_N d\mathbf{h} d\hat{\beta} d\sigma^2 \quad (4.3)$$

The optimal value for $Q(\beta_1 \dots \beta_N, \mathbf{h}, \hat{\beta}, \sigma)$ that makes the bound tight is given by the joint posterior $P(\beta_1 \dots \beta_N, \mathbf{h}, \hat{\beta}, \sigma | \mathbf{y})$ but since this posterior is intractable, we make an approximation by assuming that the posterior over the variables is independent and can be expressed as :

$$Q(\beta_1 \dots \beta_N, \mathbf{h}, \hat{\beta}, \sigma | \mathbf{y}) = \prod_i Q(\beta_i|\mathbf{y}) \prod_j Q(h_j^2|\mathbf{y}) Q(\hat{\beta}|\mathbf{y}) Q(\sigma^2|\mathbf{y})$$

This form of approximation is often called an *ensemble* variational approximation (Beal, 2003), details of which can found in Appendix A. Substituting the factorized approximation in eq. (4.3) we get:

$$\begin{aligned}
\mathcal{F}_{approx} = &\sum_i \left[\langle \ln P(y_i|\beta_i, \sigma) \rangle_{Q_{\beta_i}, Q_{\sigma^2}} + \left\langle \ln P(\beta_i|\hat{\beta}, h_1 \dots h_{d+1}) \right\rangle_{Q_{\beta_i}, Q_{h_1} \dots Q_{h_{d+1}}, Q_{\hat{\beta}}} \right] \\
&+ \sum_j \langle \ln P(h_j^2|a_j, b_j) \rangle_{Q_{h_j}} + \left\langle \ln P(\hat{\beta}|\mu, \mathbf{S}) \right\rangle_{Q_{\hat{\beta}}} + \langle \ln P(\sigma^2|c, d) \rangle_{Q_{\sigma^2}} \\
&- \sum_i \left\langle \ln Q_{\beta_i} \right\rangle_{Q_{\beta_i}} - \sum_j \langle \ln Q_{h_j} \rangle_{Q_{h_j}} - \left\langle \ln Q_{\hat{\beta}} \right\rangle_{Q_{\hat{\beta}}} - \langle \ln Q_{\sigma^2} \rangle_{Q_{\sigma^2}}
\end{aligned} \quad (4.4)$$

where $\langle . \rangle_Q$ denotes the expectation with respect to the distribution Q. The optimal values of the posterior probabilities can be computed iteratively by maximizing the functional $\mathcal{F}_{approx}$ with respect to each individual posterior distribution keeping the other distributions fixed akin to an EM procedure.

The posteriors for the parameters of RVC can be derived following the same procedure detailed above. Here, we give the resulting updates for the hyperparameters of the posterior with the detailed derivation listed in Appendix B.

- The update of the posterior $Q(\beta_i|\mathbf{y}) \sim \mathcal{N}(\nu_i, \mathbf{G}_i)$ keeping $Q(\hat{\beta}|\mathbf{y})$, $Q(h_j^2|\mathbf{y})$ and $Q(\sigma^2|\mathbf{y})$ fixed is :

$$\begin{aligned} \mathbf{G}_i &= (\mathbf{x}_i \mathbf{x}_i^T / \langle \sigma^2 \rangle + \langle C_i \rangle^{-1})^{-1} \\ &= \langle C_i \rangle - \frac{\langle C_i \rangle \mathbf{x}_i \mathbf{x}_i^T \langle C_i \rangle}{\langle \sigma^2 \rangle + \mathbf{x}_i^T \langle C_i \rangle \mathbf{x}_i} \quad \text{by Sherman-Morrison Woodbury theorem} \end{aligned} \tag{4.5}$$

  Here, $\langle \mathbf{C}_i \rangle = diag(\mathbf{x}_i^T \mathbf{x}_i / \left\langle h_j^2 \right\rangle_{Q(h_j^2)})$ and $\langle \sigma^2 \rangle = \tilde{d}/\tilde{c}$. Furthermore, using results from eq. (4.5),

$$\begin{aligned} \nu_i &= \mathbf{G}_i (y_i \mathbf{x}_i / \langle \sigma^2 \rangle + \langle C_i \rangle^{-1} \tilde{\mu}) \\ &= \frac{\langle C_i \rangle \mathbf{x}_i}{(\langle \sigma^2 \rangle + \mathbf{x}_i^T \langle C_i \rangle \mathbf{x}_i)} \left(y_i - \mathbf{x}_i^T \tilde{\mu} \right) + \tilde{\mu} \end{aligned} \tag{4.6}$$

- The update of the posterior $Q(\hat{\beta}|\mathbf{y}) \sim \mathcal{N}(\tilde{\mu}, \tilde{\mathbf{S}})$ keeping $Q(\beta_i|\mathbf{y})$, $Q(h_j^2|\mathbf{y})$ and $Q(\sigma^2|\mathbf{y})$ fixed is :

$$\tilde{\mathbf{S}} = \left(\sum_i \langle \mathbf{C}_i \rangle^{-1} + \mathbf{S}^{-1} \right)^{-1} \tag{4.7}$$

$$\tilde{\mu} = \tilde{\mathbf{S}} \left(\sum_i \langle \mathbf{C}_i \rangle^{-1} \nu_i + \mathbf{S}^{-1} \mu \right) \tag{4.8}$$

  The update given in eq. (4.8) can be seen to be equivalent to a weighted least squares regression by substituting the value of ν_i from eq. (4.6), $\tilde{\mathbf{S}}$ from eq. (4.7) in eq. (4.8) and setting $\mu = 0$ to obtain :

$$\tilde{\mu} = \left(\sum_i \langle \mathbf{C}_i \rangle^{-1} + \mathbf{S}^{-1} \right)^{-1} \left(\sum_i \frac{\mathbf{x}_i y_i}{\langle \sigma^2 \rangle + \mathbf{x}_i^T \langle C_i \rangle \mathbf{x}_i} - \left(\frac{\mathbf{x}_i \mathbf{x}_i^T}{\langle \sigma^2 \rangle + \mathbf{x}_i^T \langle C_i \rangle \mathbf{x}_i} - \langle \mathbf{C}_i \rangle^{-1} \right) \tilde{\mu} \right) \tag{4.9}$$

  which at convergence would yield :

$$\tilde{\mu} = \left(\sum_i \frac{\mathbf{x}_i \mathbf{x}_i^T}{\langle \sigma^2 \rangle + \mathbf{x}_i^T \langle C_i \rangle \mathbf{x}_i} + \mathbf{S}^{-1} \right)^{-1} \left(\sum_i \frac{\mathbf{x}_i y_i}{\langle \sigma^2 \rangle + \mathbf{x}_i^T \langle C_i \rangle \mathbf{x}_i} \right)$$

and has the same form as a weighted regression estimate with the weights given by $\left\langle \sigma^2 \right\rangle + \mathbf{x}_i^T \left\langle C_i \right\rangle \mathbf{x}_i$.

- The update of the posterior $Q(\sigma^2|\mathbf{y}) \sim \textit{Inv-Gamma}(\tilde{c}, \tilde{d})$ keeping $Q(\beta_i|\mathbf{y})$, $Q(h_j^2|\mathbf{y})$ and $Q(\hat{\beta}|\mathbf{y})$ fixed is :

$$\tilde{c} = c + N/2 \tag{4.10}$$

$$\tilde{d} = d + \sum_i \left[(y_i - \nu_i^T \mathbf{x}_i)^2 + \mathbf{x}_i^T \mathbf{G}_i \mathbf{x}_i \right] / 2 \tag{4.11}$$

- The update of the posterior $Q(h_j^2|\mathbf{y}) \sim \textit{Gamma}(\tilde{a}_j, \tilde{b}_j)$ keeping $Q(\beta_i|\mathbf{y})$, $Q(\sigma^2|\mathbf{y})$ and $Q(\hat{\beta}|\mathbf{y})$ fixed is :

$$\tilde{a}_j = a_j + N/2 \tag{4.12}$$

$$\tilde{b}_j = b_j + \sum_i \left[(\nu_{i,j} - \tilde{\mu}_{i,j})^2 + \mathbf{G}_{i,jj} + \tilde{\mathbf{S}}_{jj} \right] / (2\mathbf{x}_i^T \mathbf{x}_i) \tag{4.13}$$

Here, $\nu_{i,j}$ and $\tilde{\mu}_{i,j}$ denote the j-th element of the respective vectors and $\mathbf{G}_{i,jj}$ and $\tilde{\mathbf{S}}_{jj}$ denotes the j-th diagonal element.

These updates are repeated till convergence. The regularizer hyperparameters a_j and b_j in eqs. (4.12) and (4.13) are obtained by maximizing the bound $\mathcal{F}_{approx}$ given by eq. (4.4) with respect to these hyperparameters keeping the posterior distributions Q fixed. Such a method of hyperparameter estimation has been used previously in (Tipping & Lawrence, 2005) and suggested in (Beal, 2003). Considering only the terms of $\mathcal{F}_{approx}$ involving the hyperparameters we can define $\mathcal{E}$ as :

$$\mathcal{E} = \int Q(h_j^2|\tilde{a}_j, \tilde{b}_j) \ln P(h_j^2|a_j, b_j) dh_j^2$$

Maximizing $\mathcal{E}$ with respect to the hyperparameters is equivalent to minimizing the KL divergence between the distributions Q and P. Since the posterior Q and prior P share the same parametric form, KL divergence is minimized when the parameters of these distributions match. This leads to the simple update rule for the hyperparameters given by:

$$a_j = \tilde{a}_j, \quad b_j = \tilde{b}_j \tag{4.14}$$

The iterative training algorithm for each local model is concisely summarized in Algorithm 2.

Algorithm 2 Training a local model

1: Initialize hyperparameters: $\Theta \equiv \{\mu_0, \mathbf{S}, c, d, \mathbf{a}, \mathbf{b}\}$.
2: Input: Batch training data $\mathbf{X}$, $\mathbf{y}$
3: **repeat**
4: Estimate posterior hyperparameters $\tilde{\Theta}$ using Θ and eq. (4.5), (4.6) and eqs. (4.7) - (4.13).
5: Estimate values of the hyperparameters $\mathbf{a}$ and $\mathbf{b}$ of the regulariser prior using eq. (4.14).
6: **until** convergence of $\tilde{\Theta}$

4.2 Prediction using the committee of local models

We have dealt so far with building a coherent probabilistic model for each local expert and have derived inference procedures to estimate the parameters of individual models. Given the ensemble of trained local models, in order to predict the response y_q for a new query point $\mathbf{x}_q$, we take the normalized product of the *predictive distribution* of each local model as discussed in Chapter 2. The predictive distribution of each local model is given by:

$$P(y_q|\mathbf{y}) = \int P(y_q|\hat{\beta}, \sigma, \mathbf{h}) Q(\hat{\beta}|\mathbf{y}) Q(\sigma^2|\mathbf{y}) Q(\mathbf{h}|\mathbf{y}) d\mathbf{h} d\hat{\beta} d\sigma^2 \tag{4.15}$$

where $P(y_q|\hat{\beta}, \sigma, \mathbf{h})$ has the form given by eq. (3.6). We can further integrate out $\hat{\beta}$ from eq. (4.15) to obtain the predictive distribution as :

$$P(y_q|\mathbf{y}) = \int P(y_q|\tilde{\mu}, \tilde{\mathbf{S}}, \sigma, \mathbf{h}) Q(\sigma^2|\mathbf{y}) Q(\mathbf{h}|\mathbf{y}) d\mathbf{h} d\sigma^2 \tag{4.16}$$

Eq. (4.16) is a mixture of Gaussians with a common mean and different variances. Since the integral in eq. (4.16) is intractable, it is approximated using a Gaussian distribution given by $\mathcal{N}(\tilde{\mu}^T \mathbf{x}_q, v^2)$. A student-t distribution would be a better approximation for the particular integral, but we have chosen to approximate the predictive distribution by a Gaussian keeping in mind the ease of combining Gaussian distributions using a product of experts paradigm. By minimizing the KL divergence between the approximate distribution and the actual distribution, it can be shown (Appendix C) that the optimal value of v^2 is $\mathbf{x}_q^T (\tilde{\mathbf{S}} + C_{\mathbf{h}_{mode}}) \mathbf{x}_q + \sigma^2_{mean}$ where $C_{\mathbf{h}_{mode}}$ refers to the covariance matrix evaluated at the mode of the posterior $Q(\mathbf{h}|\mathbf{y})$ and σ^2_{mean} stands for the mean of $Q(\sigma^2|\mathbf{y})$. The final predictive distribution for the k-th local model is:

$$y_{q,k} \sim \mathcal{N}(\tilde{\mu}_k^T \mathbf{x}_{q,k}, \mathbf{x}_{q,k}{}^T (\tilde{\mathbf{S}}_k + C_{k\mathbf{h}_{mode}}) \mathbf{x}_{q,k} + \sigma^2_{mean}),$$

Algorithm 3 Global prediction using local models

1: Input: Query point $\mathbf{x}_q$, learned parameters $\tilde{\mu}_k$, C_k, $\tilde{\mathbf{S}}_k$ and σ_k
2: Initialize: $sum_\alpha = 0$, $y_q = 0$
3: **for** k = 1 to #local models **do**
4: $\quad \mathbf{x}_{q,k} = \mathbf{x}_q - \mathbf{x}_{c,k}$
5: $\quad$ Calculate α_k using eq. (4.18)
6: $\quad y_q = y_q + \alpha_k \tilde{\mu}_k^T \mathbf{x}_{q,k}$
7: $\quad sum_\alpha = sum_\alpha + \alpha_k$
8: **end for**
9: Output: $y_q = y_q / sum_\alpha$, $var = 1/sum_\alpha$

where $\mathbf{x}_{q,k}$ refers to the query point with the k-th center subtracted and augmented with bias. Blending the prediction of different experts by taking their product and normalizing it results in a Normal distribution given by :

$$y_q \sim \mathcal{N}(\mu, \zeta^2) \quad \text{where} \quad \mu = \frac{\sum_k \alpha_k \tilde{\mu}_k^T \mathbf{x}_{q,k}}{\sum_k \alpha_k}, \;\; \zeta^2 = \frac{1}{\sum_k \alpha_k}. \tag{4.17}$$

Here, μ is a sum of the means of each individual expert weighted by the confidence expressed by each expert in its own prediction α_k, ζ^2 is the variance and α_k is the precision of each expert :

$$\alpha_k = 1/(\mathbf{x}_{q,k}^T (\tilde{\mathbf{S}}_k + C_k)\mathbf{x}_{q,k} + \sigma_k^2), \;\; C_k = diag\{\mathbf{x}_{q,k}^T \mathbf{x}_{q,k} / h_{j,k}^2\} \tag{4.18}$$

The prediction routine for RVC combining the predictions of individual local models is summarized in Algorithm 3.

4.3 Empirical study of RVC

In this section, we carry our some empirical evaluations of the RVC algorithm in order to highlight the salient aspects of its formulation. In the implementation, the hyper-parameters μ, $\mathbf{S}$, c and d are set to values that make the corresponding priors non-informative. Values of $\mu = \mathbf{0}$, $\mathbf{S} = 10^3 \times \mathbf{I}$, $c = 10^{-3}$ and $d = 10^{-3}$ ensures such a condition. On the other hand the regularizer hyperparameters $\mathbf{a}$ and $\mathbf{b}$ are initialized such that it encourages small $\mathbf{h}$ as previously discussed in Section 3.2.2. A value of $\mathbf{a} = 1$ and a sufficiently large value for $\mathbf{b}$ ensures such a bias. These settings are used for all the evaluations in the thesis.

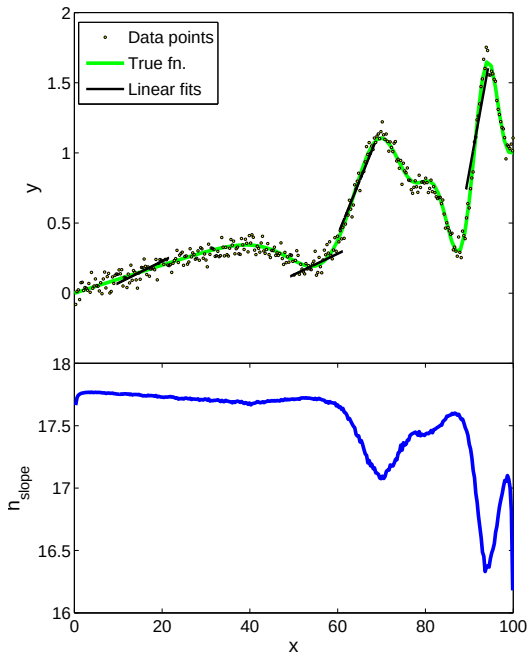


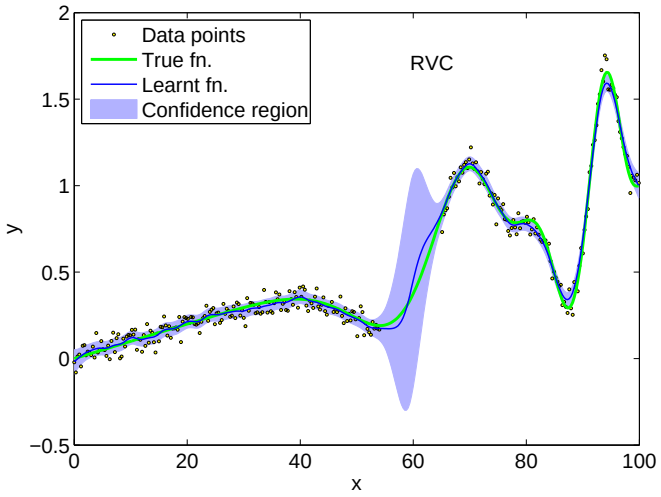
Figure 4.1: Local linear fits and 'local' bandwidth adaptation

4.3.1 Bandwidth adaptation and confidence measures

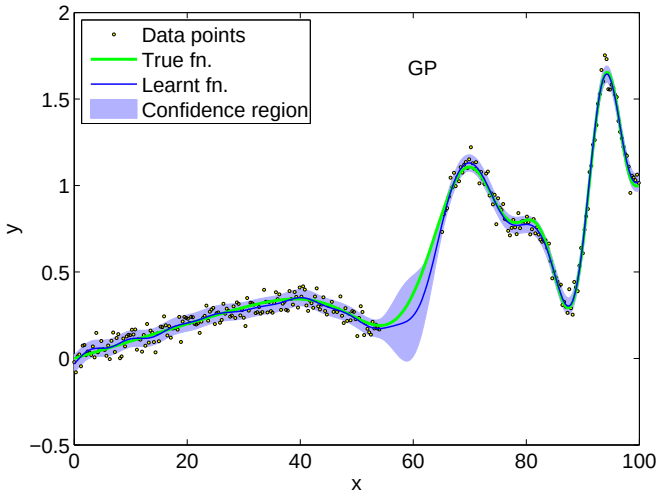
The bandwidth parameter plays a crucial role in determining the bias-variance tradeoff for each local model by determining the extent of each local model. Here, we illustrate the adaptation of bandwidth for each local model and the resulting global fits and confidence bounds of the RVC model. Fig. 4.1 shows the local linear fits (at selected test points) learnt by RVC using 300 noisy training data points from a function with varying spatial complexity(Vijayakumar et al., 2005). Such functions are extremely hard to learn since models with high bias tends to oversmooth the nonlinear regions while more complex models tend to fit the noise. While the local linear fits roughly correspond to the expected slope at the respective points (see Fig. 4.1 (top)), avoiding overfitting or oversmoothing, a more significant result is the adaptation of the local bandwidth. The bottom section of Fig. 4.1 plots the converged locality measure[1] in terms of the local bandwidth parameter h_{slope}, illustrating the adaptation to the local curvature.

In another evaluation using the same toy function and 500 noisy training points, we compared the global fit and confidence bounds learned by RVC to Gaussian Process

[1]Smooth plots for the bandwidth were obtained by using local models centered around a dense uniform grid in input space.



(a) Fit and confidence bounds learnt by RVC



(b) Fit and confidence bounds learnt by GP

Figure 4.2: Comparison of fits learnt by RVC and GP

(GP) regression (Williams, 1998) – a state of the art regression technique. Fig. 4.2(a) and 4.2(b) shows the results of the comparison; it is important to note that we have deliberately avoided using training data in the input data range [55,65] and the confidence bounds (computed using eqs. (4.17) and (4.18) for RVC) nicely reflect this.This evaluation is aimed to demonstrate that RVC is able to adapt its confidence bounds based on the distribution of training data as well as a global learning algorithm like GP. This comparison does not include the fit learned in the region of sparse data, since the lack of data precludes any judgement about the true fit.

This evaluation highlights the fact that RVC provides competitive (with respect to

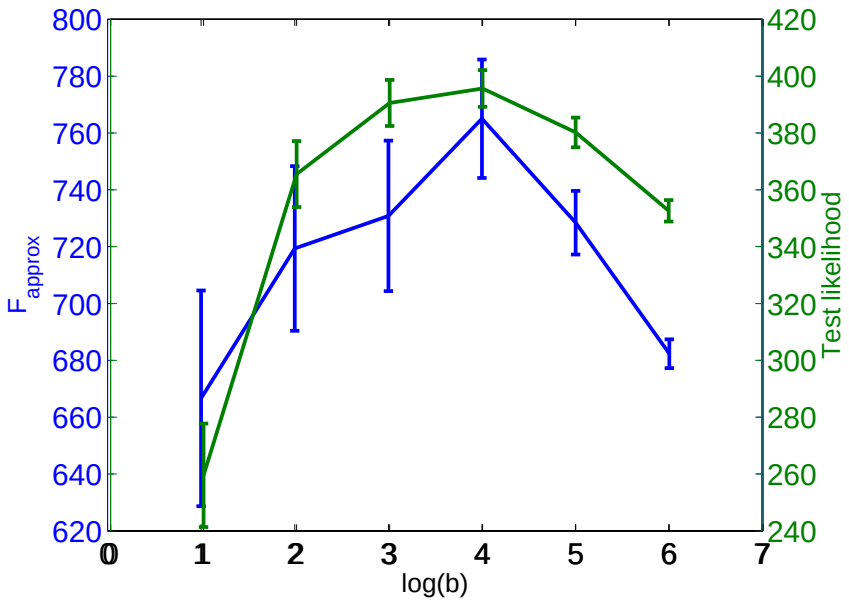


Figure 4.3: Sensitivity to initialization of the hyperparameters

the state of the art), approximate and appropriate confidence interval estimates while retaining the attractive properties of non-parametric, localised learning.

4.3.2 Sensitivity to initialization

Although the Variational Bayesian EM(VBEM) procedure of learning in RVC is not entirely parameter free, the majority of the parameters and hyperparameters (refer to Fig. 3.4), with the exception of the scale parameter of the Gamma regularizer hyperprior **b**, can be set using uninformative priors – we will demonstrate this empirically later on in this section. Maximum likelihood estimation of the regularizer hyperparameter **b** makes the convergence of the RVC relatively sensitive to its initialization. Next, we explore this effect on the final learned model and recommend a procedure to determine the optimal initialization for **b**.

As an illustrative example, a noisy (std. deviation of 0.05) *sinc* function with 200 randomly distributed training points was used to train RVC. The training was repeated for different values of the scale parameter **b** of the regularizer Gamma prior. Fig. 4.3 plots the predictive likelihood of test data and the value of $\mathcal{F}_{approx}$ as defined by eq. (4.4) after convergence of learning. These quantities are plotted against different initializations of **b** measured on a log scale. From the figure (and further extensive evaluations), it can be gleaned that while different initializations can lead to different points of

convergence in the parameter space, the variation of the converged value of the objective function($\mathcal{F}_{approx}$) closely mimics that of the test likelihood. Hence, this objective function can be used as a reliable measure to choose a near-optimal initialization for the hyperparameter **b**.

In contrast to the scale hyperparameter **b**, for the parameters of fit ($\hat{\beta}$) and noise variance (σ^2) – again refer Fig. 3.4, it is sufficient to use uninformative priors over them. This insensitivity can be illustrated via an example where the role of prior in learning the noise parameter σ is studied. An experiment was performed where the evolution of test likelihood was traced under three different settings of prior over the parameter σ. The value of σ enters into the learning updates through the expectation statistic $\left\langle \sigma^2 \right\rangle$ which in turn is determined by the updates of $\tilde{d}$ and $\tilde{c}$ in eqs. (4.10) and (4.11), and is given by :

$$\left\langle \sigma^2 \right\rangle = \frac{\tilde{d}}{\tilde{c}} = \frac{d_{prior} + d_{data}}{c_{prior} + c_{data}} \tag{4.19}$$

where $d_{data} = \sum_i \left[(y_i - \nu_i^T \mathbf{x}_i)^2 + \mathbf{x}_i^T \mathbf{G}_i \mathbf{x}_i \right] / 2$, $c_{data} = N/2$, $c_{prior} = c$ and $d_{prior} = d$. Eq. (4.19) can then be rewritten as :

$$\left\langle \sigma^2 \right\rangle = \frac{\left\langle \sigma^2_{prior} \right\rangle / c_{data} + \left\langle \sigma^2_{data} \right\rangle / c_{prior}}{1/c_{data} + 1/c_{prior}} \tag{4.20}$$

where $\left\langle \sigma^2_{prior} \right\rangle = \frac{d_{prior}}{c_{prior}}$ and $\left\langle \sigma^2_{data} \right\rangle = \frac{d_{data}}{c_{data}}$. It is evident from eq. (4.20) that the posterior estimate is a weighted average of the data estimate ($\sigma^2_{data}, c_{data}$) and the prior ($\sigma^2_{prior}, c_{prior}$). The relative contribution of prior and data in determining the posterior can be modulated by adjusting the ratio of c_{prior} to c_{data}. The learning behavior of RVC in terms of accuracy and convergence is studied under three different priors for σ^2:

- Correct informative prior : In this setting, $\left\langle \sigma^2_{prior} \right\rangle$ is initialized to the true noise value used in generating the dataset and $c_{prior}/c_{data} = 5$. This setting provides a higher weight to the prior belief.

- Non-informative prior : In this setting, $c_{prior} = 10^{-3}$ and $d_{prior} = 10^{-3}$ which corresponds to a $\left\langle \sigma^2_{prior} \right\rangle = 1$ and $c_{prior}/c_{data} = 5 \times 10^{-6}$. This setting corresponds to a case where the prior is downweighted and allows the "data to speak" (Gelman et al., 2003).

- Incorrect informative prior : In this setting, $\left\langle \sigma^2_{prior} \right\rangle$ is initialized with a random incorrect value and $c_{prior}/c_{data} = 5$. This setting corresponds to a case where the prior belief is incorrect but is allowed to dominate over the data.

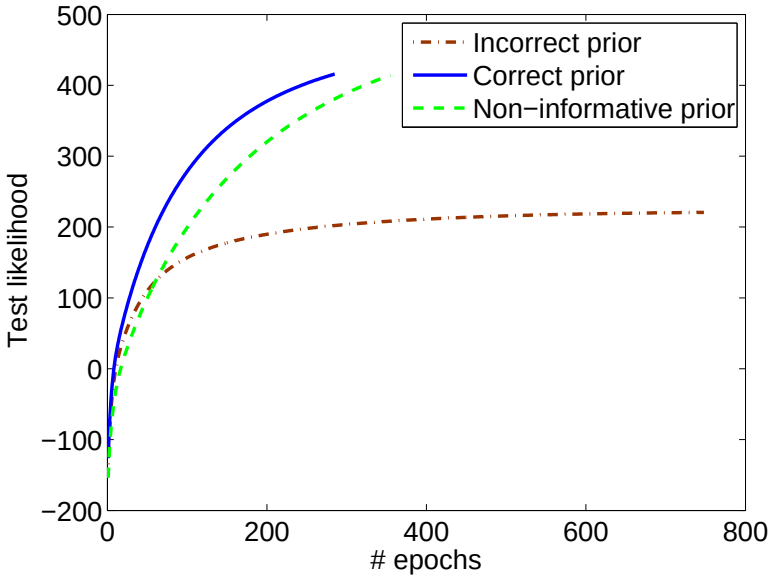


Figure 4.4: Effect of prior on accuracy and convergence

The plots of test likelihood for these settings is shown in Fig. 4.4. As can be seen from the figure, a non-informative setting of the prior leads to the same likelihood as a correct prior albeit at a slower rate of convergence whereas an incorrect strong prior leads to a deterioration in accuracy accompanied with a slower convergence. This suggests that in the absence of any strong prior information about the values of the parameters, a non-informative prior serves as the best choice; indeed, these default settings can be carried over to different learning tasks without the need for extensive tuning. The resulting reduction in the number of open parameters in RVC is a significant improvement over other local linear regression methods like LWPR (Vijayakumar et al., 2005).

4.3.3 Allocation of local models

Modularity and independence of local models lets us adapt the complexity of the combined model by the addition and deletion of local models without affecting the existing learnt models. This property of RVC makes it useful as an adaptive, incremental learner that maintains the key benefits of local learning (Schaal & Atkeson, 1998; Vijayakumar et al., 2005) and will be exploited in Section 5.2, wherein local models are allocated and deallocated dynamically. There are two issues when dealing with the allocation of local models - the number and the position of these models in the input space. In a *non-competitive* local learning scenario, where local models cooperate only

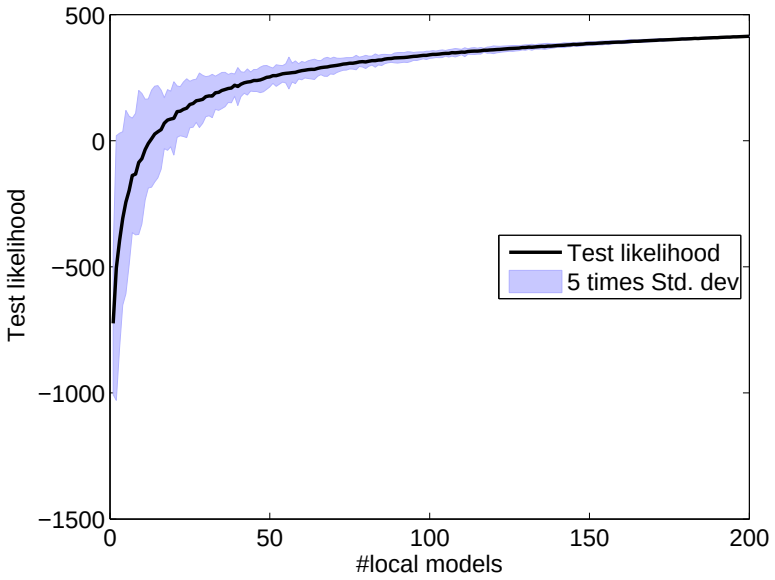


Figure 4.5: Illustration of the effect of number of local models on the quality of fit

at the prediction phase, the exact position and number of local models do not affect the quality of approximation(Atkeson et al., 1997; Schaal & Atkeson, 1998) as long as sufficient overlap is maintained.

To illustrate this, noisy (std. deviation of 0.05) training data from a *sinc* function is used and accuracy of RVC is measured as a function of the number of local models used. To make the evaluation efficient, we exploit the fact that the local models are trained independently. Hence, instead of retraining RVC models with different number of local models, local models are placed at all the training data points and trained. A subset of these local models is then chosen to build an RVC for prediction. The RVC built in this manner is used to compute the test likelihood over a test dataset. This process of prediction is repeated for RVC built with a single local model, two local models and so on upto the maximum number of local models available. For each cardinality of the RVC, mean and variance of test likelihood is computed by repeating the test with different random selections of local models from the pool of trained local models. The statistics thus collected is shown in Fig. 4.5. From the figure, it is clear that after a certain point there is no significant increase in the accuracy with further addition of local models. Also, as seen from a larger variance at low number of local models and a much lower variance when using larger numbers of local models, the exact positioning of the local models is significant only when the number of local models are low, i.e., when there is not sufficient overlap between models. These results

are indeed in line with other more extensive studies on local weighted learning(Atkeson et al., 1997).

4.4 Discussion

In this chapter we have formulated a principled variational approximation to the posterior probabilities over parameters of the model. The EM optimization procedure built on top of the variational Bayesian approximation allows us to develop efficient updates for the posterior probabilities of the parameters of the local model. The most interesting aspect of the learning rules for the local models is that they are completely independent of each other and hence avoid negative interference between them.

Training local models is in itself not sufficient to make predictions for novel test points and requires a combination rule to combine the predictions of the local models. In this chapter we formulated such a combination rule based on the framework of product of experts. Furthermore, through empirical analysis of the RVC model in Section 4.3 we demonstrated the properties of the learning algorithm used in RVC. To summarise, the outcome of this chapter is a learning algorithm that has a probabilistic formulation and uses a Bayesian inference procedure to learn the posterior distributions over its parameters, from a *batch* of data. In the next chapter we modify this algorithm to make it online.

Chapter 5

Online learning

In previous chapters, we have looked at the probabilistic formulation of RVC and derived learning rules that involved learning the posterior probabilities of the parameters using a batch update represented by eqs. (4.7)-(4.13). These updates can be rewritten in the form of online updates by exploiting the Bayesian formalism (Sato, 2001; Opper, 1998). In a batch mode of posterior evaluation, we have

$$posterior_N = \prod_{i=1}^{N} (likelihood_i) \times prior_0$$

where i is an index over the data points. The same can be expressed as a set of online updates:

$$\begin{aligned} posterior_i &\propto likelihood_i \times prior_i \\ prior_{i+1} &= posterior_i \end{aligned}$$

This set of updates implies that at every step of the online update the prior computed over the data seen so far is combined with the likelihood of the current data point to yield the posterior. This new posterior distribution of the parameter is then used as the prior during the next update. Based on this, we can derive the online updates for RVC that correspond to the batch results derived earlier :

$$\tilde{\mathbf{S}}_i = (\langle \mathbf{C}_i \rangle^{-1} + \mathbf{S}_i^{-1})^{-1} \tag{5.1}$$

$$\tilde{\mu}_i = \tilde{\mathbf{S}}_i (\langle \mathbf{C}_i \rangle^{-1} \nu_i + \mathbf{S}_i^{-1} \mu_i) \tag{5.2}$$

$$\tilde{a}_{i,j} = a_{i,j} + 1/2 \tag{5.3}$$

$$\tilde{b}_{i,j} = b_{i,j} + \left[(\nu_{i,j} - \tilde{\mu}_{i,j})^2 + \mathbf{G}_{i,jj} + \tilde{\mathbf{S}}_{i,jj} \right] / (2\mathbf{x}_i^T \mathbf{x}_i) \tag{5.4}$$

$$\tilde{c}_i = c_i + 1/2 \tag{5.5}$$

$$\tilde{d}_i = d_i + \left[(y_i - \nu_i^T \mathbf{x}_i)^2 + \mathbf{x}_i^T \mathbf{G}_i \mathbf{x}_i \right] / 2 \tag{5.6}$$

Algorithm 4 Online updates for a local model

1: Initialize hyperparameters: $\Theta_0 \equiv \{\mu_0, \mathbf{S}_0, c_0, d_0, \mathbf{a}_0, \mathbf{b}_0\}$.
2: **repeat**
3: Input training data: $\mathbf{x}_i$, y_i
4: **repeat**
5: Estimate posterior hyperparameters $\tilde{\Theta}_i$ using Θ_i and eq. (4.5), (4.6) and eqs. (5.1) - (5.6).
6: Estimate values of the hyperparameters **a** and **b** of the regulariser prior using eq. (4.14).
7: **until** convergence of posteriors
8: $\Theta_{i+1} = \tilde{\Theta}_i$
9: **until** end of training data

We repeat the above updates for a single data point $\{\mathbf{x}_i, y_i\}$ till the posteriors $\tilde{\Theta} = (\tilde{\mathbf{S}}, \tilde{\mu}, \tilde{a}, \tilde{b}, \tilde{c}, \tilde{d})$ converge. For the $(i+1)$-th point, we then use posterior $\tilde{\Theta}$ of i-th step as the prior $\Theta = (\mathbf{S}, \mu, \mathbf{a}, \mathbf{b}, c, d)$; this procedure is illustrated in Algorithm 4. The prediction subroutine for the online algorithm remains the same as shown in Algorithm 3.

5.1 Complexity analysis of online updates

The time complexity of the learning algorithm is dominated by the computation of $\mathbf{G}_i$ in eq. (4.5). The equations that use $\mathbf{G}_i$ are eq. (5.4) and eq. (5.6) and these can be rewritten to avoid explicit computation of $\mathbf{G}_i$. Eq. (5.4) requires only the diagonal elements of $\mathbf{G}_i$ which can be computed in $O(d)$(d is the number of dimensions of the input) since

$$\mathbf{G}_i(j,j) = \mathbf{C}_i(j,j) - (\mathbf{C}_i(j,j)\mathbf{x}_i(j))^2/(\sigma^2 + \gamma_i) \quad \text{using eq. (4.5)}$$

where $\gamma_i = \mathbf{x}_i^T \mathbf{C}_i \mathbf{x}_i$ which can also be computed in $O(d)$ due to the fact that $\mathbf{C}_i$ is diagonal. On the other hand, eq. (5.6) requires the evaluation of $\mathbf{x}_i^T \mathbf{G}_i \mathbf{x}_i$ which in turn can be written down as:

$$\mathbf{x}_i^T \mathbf{G}_i \mathbf{x}_i = \frac{\sigma^2 \gamma_i}{\sigma^2 + \gamma_i}$$

and can also be computed in $O(d)$. Furthermore, the matrix inverses in eq. (5.1) and eq. (5.2) can also be computed in $O(d)$ due to the fact that $\mathbf{S}_i$ and $\mathbf{C}_i$ are diagonal matrices. Therefore, the overall time complexity per online update is $O(dM)$ where d is the number of dimensions and M the number of local models. The algorithm

does not require any data points to be stored and hence, has a $O(M)$ space complexity for the sufficient statistics stored in the local models. The independence of the local models also means that the effective time complexity can be brought down to $O(d)$ using M parallel processors. The time complexity for prediction is $O(dM)$ including the evaluation of mean and the confidence bounds. We can see from this analysis that the algorithm is very efficient with respect to time and space (in fact it matches LWPR's efficiency) and hence, is a strong candidate for situations which require real time and online learning.

5.2 Addition/deletion of local models

An important issue in online learning is the dynamic allocation of local models. It has been shown in Section 4.3.3 that typically only a subset of the training points needs to be used as centers of local models. Finding such an optimal subset of the training points, in a *real time* and *independent* learner is difficult and non-essential. The motivation behind this statement can be summarized as:

- In an online setting, at any point in time there is only a partial knowledge of the target function. Hence, in theory, the optimal centers for the local models could shift with time as more samples are provided to the learner. This is particularly true when we have a changing input data distribution (also known as covariate shift) – a scenario which is very realistic in, for e.g., robot dynamics learning where data arrives in batches from one part of the workspace and then, another as opposed to a uniform distribution.

- The optimal centers for the local models is a function of the global fit and cannot be determined locally and independent of other models. Moreover, truly incremental learning schemes discard training data after incorporating it into the learning model. Indeed, Section 4.3.3 has shown that sufficient overlap between local models is enough to ensure good global prediction.

Hence, the local models are added using a simple heuristic based on the predictive likelihood for a new data point. The predictive likelihood for an unseen data point measures the capacity of the existing local models to explain the new data point. If the likelihood is less than a threshold parameter for all local models, then a new local model is added with the data point serving as the center.

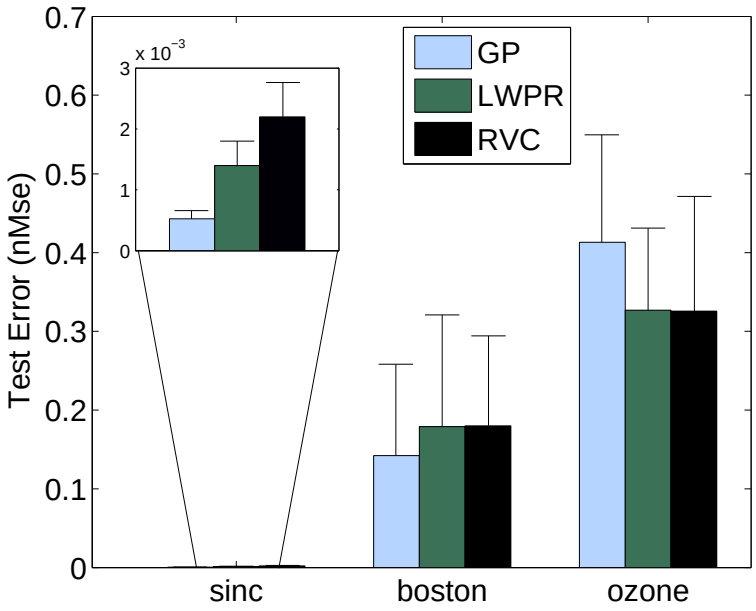


Figure 5.1: Comparison of generalization error between RVC, GP and LWPR on sinc, boston and ozone data sets.

When two local models have sufficient overlap in the region they model, then one of them is redundant and can be pruned. The degree of overlap between two models can be estimated by looking at the input region where these two models have a significantly high confidence. If the two candidate models express confidence for a prediction above a fixed threshold then it is likely that the models have a significant overlap and one of them can then be pruned. In empirical studies though, we have found that if the addition of models is done judiciously then pruning does not play a significant part in the learning process. The addition and deletion heuristics used here are similar to the ones used in (Schaal & Atkeson, 1998).

5.3 Evaluation

In this section, we demonstrate the salient aspects of the RVC model by looking at some empirical test results, compare the accuracy and robustness against state of the art methods and evaluate its performance on some benchmark datasets.

In the first experiment we compare the generalization performance of RVC against two other candidate algorithms - Gaussian Process(GP)(Williams, 1998) and Locally Weighted Projection Regression(LWPR)(Vijayakumar et al., 2005) on artificial as well

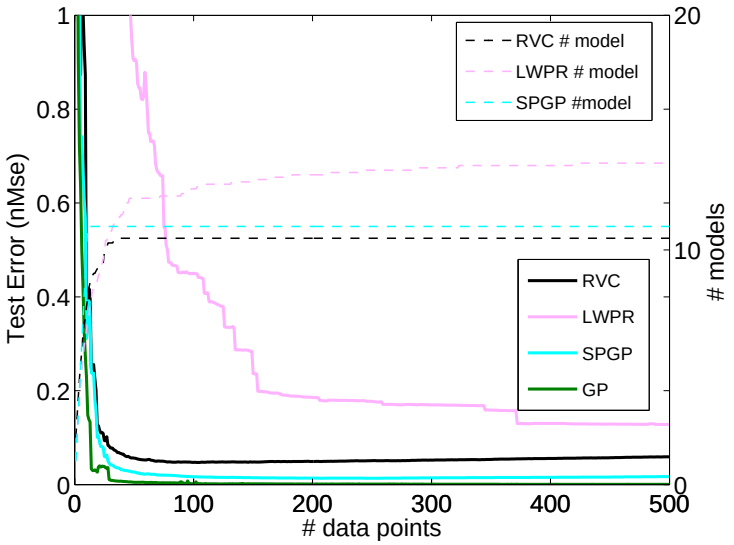
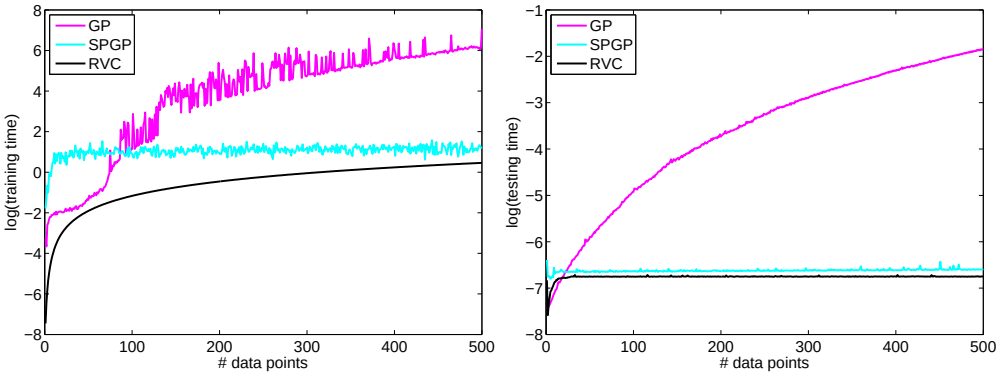


Figure 5.2: Comparison of online learning dynamics for *sinc* function

as real world datasets. The *sinc* function, air dataset described in (Bruntz et al., 1974) and the Boston housing dataset from the UCI repository were used as benchmark datasets. The dataset for the *sinc* function consisted of 500 training data points from the *sinc* function corrupted with output noise: $\varepsilon \sim \mathcal{N}(0, 0.05^2)$ and a test dataset consisting of 1000 uniformly distributed test points. The air(ozone) dataset which is a three dimensional dataset with 111 data points was split into 83 training and 28 test points. The 13 dimensional Boston dataset was split into 404 training and 102 test points. The online learning methods – RVC and LWPR, were trained in epochs of repeated presentation of the training data, till convergence; LWPR required careful tuning of the distance metric initialization and learning rates to achieve the performance reported here as opposed to the uninformative priors used for RVC. The performance of GP, RVC and LWPR shown in Fig. 5.1 are statistics accumulated over 10 different train-test splits. Asymptotically, all three methods perform very well on the *sinc* data set, achieving *nMSE* of less than 0.0025. For the ozone dataset which is highly nonlinear, RVC and LWPR performs better than GP. For the Boston dataset, we find that the performance of RVC is close to that of LWPR while slightly inferior to the GP results – although this difference is statistically insignificant.

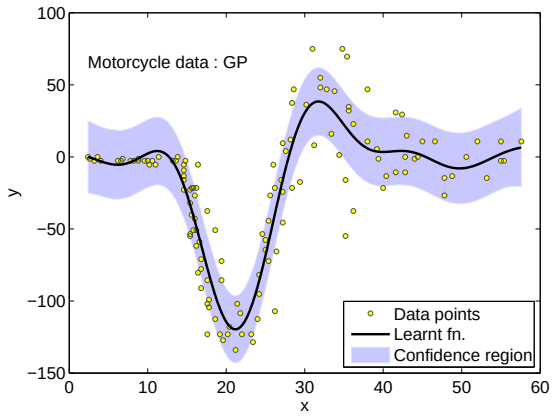
In an evaluation to compare the online learning characteristics, we trained RVC, LWPR, GP and Sparse Pseudo-input Gaussian Process (SPGP) (Snelson & Ghahramani, 2006) algorithms on the *sinc* dataset described earlier. After each training data



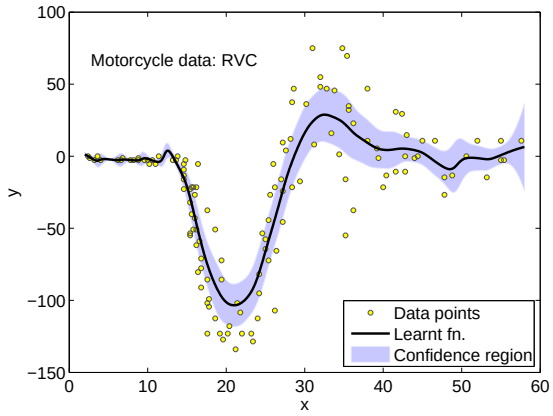
(a) (b)

Figure 5.3: (a) Comparison of training time between RVC, GP and SPGP on sinc data set. (b) Comparison of time complexity for prediction.

point was presented to the learning system, the generalization error was evaluated using a uniform grid of 500 test data points. The RVC model was allowed only a single EM iteration for each data point to ensure a fair comparison with LWPR. The resulting error dynamics averaged over 10 train-test evaluations, is shown in Fig. 5.2. In this comparison, GP exhibits a sharply decreasing error curve which is not surprising considering that it is essentially a *batch* method and stores away all of the training data for prediction. When we compare RVC with LWPR, we find that RVC converges faster while using roughly the same order of local models. This can be attributed to the Bayesian learning rules of RVC that estimates the posterior over parameters rather than point estimates. Since the posterior is a product of likelihood and prior, in the event of sparse data (as in the initial stages of online learning), the prior ensures that the posterior distributions assigned to the parameters and in turn the predictions of the learner are reasonable. Also the optimization of the regularizer hyperparameters for every data point implies a faster adaptation and hence, a faster convergence. We can now compare accuracy and efficiency of RVC with that of a sparse version of GP namely SPGP. RVC is able to match the generalization performance of an SPGP using almost the same number of local models. A more significant result is seen from Fig. 5.3(a) where the time taken for training on the *sinc* dataset is illustrated for RVC and compared with an accurate yet slow full Gaussian Process and an approximate yet fast algorithm of SPGP. The difference between RVC and SPGP is very clear from Fig. 5.3(a) with RVC having a better scaling behaviour. The empirical results supports the theoretical prediction of a $O(MN)$ time complexity for RVC compared to $O(M^2N)$



(a) Fit and confidence bounds for GP



(b) Fit and confidence bounds for RVC

Figure 5.4: Comparison of fit and confidence bounds for the motorcycle dataset learned by GP and RVC model, showing adaptation to heteroscedastic (varying) noise levels. The data consists of a sequence of accelerometer readings through time following a simulated motor-cycle crash.

for SPGP and $O(N^3)$ for a full GP. The time required for the mean prediction are the same for RVC and SPGP - $O(M)$ per test case and is much efficient than a full GP with $O(N^3)$ complexity. These experiments thus demonstrate the favourable scaling characteristics of RVC when learning from data.

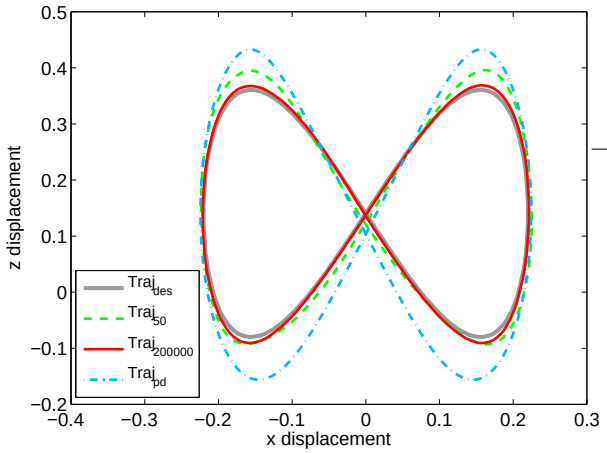
Our next experiment aims to illustrate the ability of RVC to model heteroscedastic data (i.e., data with varying noise levels). Fig. 5.4(b) illustrates the fit and the confidence interval learnt on the *motorcycle impact* data (Silverman, 1984) discussed in (Rasmussen & Gharamani, 2000). Notice that the confidence interval correctly adapts to the varying amount of noise in the data as compared to the confidence interval learnt by a GP with squared exponential kernel shown in Fig. 5.4(a). This ability to model

(a) (b)

Figure 5.5: (a) DLR LWR-III robot arm, (b) OpenGL/ODE based full contact simulation of DLR robot arm

non-stationary functions is another advantage of RVC's localised learning.

In the next evaluation, we demonstrate the utility of RVC in learning the inverse dynamics of a 7DOF *KUKA Light-Weight Robot (LWR)* arm (see Fig. 5.5(a)). Out of the seven degrees of freedom, three were controlled and the rest fixed. A batch train-test dataset consisting of 9 input dimensions and an output was generated using a faithful simulation (see Fig. 5.5(b)) of the robot arm performing a pseudo-randomly drifting figure-8 pattern. The input dimensions were the joint angles, velocities and accelerations whereas the output was the applied torque at the shoulder joint. A set of 1000 training and test points were used to evaluate the normalized mean square error (nMSE) and the performance averaged over 5 repeats of the evaluation. The performance of RVC compared against GP and LWPR is shown in Fig. 5.6(inset table). As can be seen from the table, for data from one of the representative joints of the robot arm, RVC and LWPR have similar performance and is significantly better than a simple linear regressor whereas GP has the best performance. This difference can be attributed to the fact that while GP utilizes the whole of the training set for prediction, RVC and LWPR is more efficient and uses only a few parameters for prediction. Finally, we used the robot arm as mentioned in the previous experiment to track a figure-8 pattern. A *composite* control consisting of a combination of feedforward and low gain feedback commands was used to control the joints of the arm. The feedforward command in turn was produced from an inverse dynamics model of the robot arm learnt online by an RVC model. The aim of the experiment was to learn an accurate model for the inverse dynamics of the robot arm. The RVC model was trained on data that was collected while the robot arm performed a drifting 8 pattern. The training data consisted of 2000 data points corresponding to a 8 pattern in the x-z plane and a slowly drifting y



Methods	Test error (nMse)
Linear	$\mathbf{1.4 \times 10^{-2}}(\pm 6.13 \times 10^{-4})$
GP	$\mathbf{1.45 \times 10^{-7}}(\pm 8.09 \times 10^{-9})$
RVC	$\mathbf{1.8 \times 10^{-3}}(\pm 2.19 \times 10^{-4})$
LWPR	$\mathbf{1.1 \times 10^{-3}}(\pm 3.63 \times 10^{-5})$

Figure 5.6: Progress of online learning with RVC control - $Traj_{des}$ is the desired trajectory, $Traj_{50}$, $Traj_{200000}$ are the trajectories learnt from 50 and $20,0000$ training data points and $Traj_{pd}$ is the trajectory corresponding to a low gain PD control without any feedforward component; (table) Comparison of test errors for different learning methods.

coordinate. The training was stopped at certain intervals and the robot arm with the composite control was used to track the figure-8 pattern on a fixed x-z plane. The result of the experiment is shown in Fig. 5.6 where the pattern performed by the robot is shown at different stages of learning. As can be seen from the figure, RVC model is able to achieve a significant improvement over a simple Proportional-Derivative (PD) controller within a single pass of the training data, but requires more learning iterations to accurately track the desired trajectory. In this experiment we have compared the performance of the composite control based tracking to the PD controller to illustrate the suitability of an online learner like RVC for such control applications.

5.4 Automatic relevance determination in high dimensional input space

One of the impediments of localised learning is that, with increasing dimensions of the input space, learning the locality becomes difficult and requires a dense sampling of data in the input space. As explained and empirically demonstrated in (Vijayakumar et al., 2005; Hoffmann et al., 2009), the usual methods of dimensionality reduction like PCA is insufficient to handle this problem since PCA takes into account only the correlation amongst the input variables and ignores the information provided by the

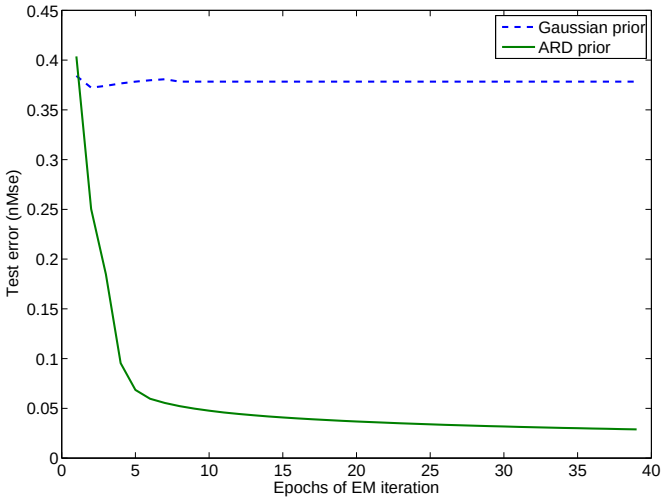


Figure 5.7: Comparison of the convergence of learning using a Gaussian prior and an ARD prior on a high dimensional dataset

response variable when determining the optimal low dimensional representation of the input space. The dimensionality reduction is more effective if the response variable is also included in the dimensionality reduction process. Consequently in this section we look at a method to reduce the dimensionality of the input space using a Automatic Relevance prior. It must be noted that while using Automatic Relevance prior allows us to choose the features it does not create new features like a PCA and so is less powerful, yet from the empirical evaluations performed later in this section we find that feature selection greatly improves the rate of convergence of the learning algorithm.

The main advantage of having a Bayesian formulation for learning is that we can integrate our prior belief into the model by changing the prior distribution over parameters. When we need to handle large number of input dimensions with some of them being irrelevant, it is efficient to identify and eliminate the irrelevant dimensions. It is especially useful to have an automatic relevance determination in RVC because of its use of distance measures defined on the input space to measure the locality of a model. When we eliminate the irrelevant bandwidth parameters from the model, the learning becomes more efficient.

We can achieve an automatic relevance determination by using a sparsity encouraging prior similar to (Figueiredo, 2003). In the original model illustrated in Fig. 3.4, we change the prior over the regression coefficient $\hat{\beta}$ from a zero mean Gaussian to a zero mean Laplacian distribution leading to -

$$P(\hat{\beta}|\alpha) \propto exp(-\alpha\|\hat{\beta}\|_1)$$

where $\|\hat{\beta}\|_1$ denotes the l_1 norm and α is the scale parameter of the Laplace distribution. We can use this prior to find the posterior distribution over $\hat{\beta}$. However the Laplace prior is not a conjugate distribution of the Gaussian distribution and requires approximation inorder to evaluate the posterior. We approximate the posterior over $\hat{\beta}$ by a Gaussian distribution using a Laplace approximation. In our case the log posterior over $\hat{\beta}$ is given by :

$$\mathcal{M} = \ln Q(\hat{\beta}|\mathbf{y}) \propto \left\langle \ln P(\beta_i|\hat{\beta}, \mathbf{C}_i) \right\rangle_{Q(\beta_i),Q(\mathbf{h})} + \ln P(\hat{\beta}|\alpha) \tag{5.7}$$

Laplace approximation of the posterior corresponds to

$$Q(\hat{\beta}|\mathbf{y}) \sim \mathcal{N}(\tilde{\mu}, \tilde{\mathbf{S}})$$

where $\tilde{\mu} = argmax_{\hat{\beta}} \mathcal{M}$ and $\tilde{\mathbf{S}}^{-1} = -\nabla\nabla\mathcal{M}|_{\hat{\beta}=\tilde{\mu}}$ is the Hessian of the negative log posterior. The posterior mode $\tilde{\mu}$ can be obtained by setting the gradient of $\mathcal{M}$ to zero and when $\mathbf{C}$ is diagonal we can evaluate the value of μ for each dimension separately:

$$\tilde{\mu}_j = sgn(\mathbf{v}_j)\left(|\mathbf{v}_j| - \frac{\alpha}{\sum_i 1/\mathbf{C}_i(j,j)}\right)_+ \tag{5.8}$$

where $sgn(.)$ is the sign of the argument, $(.)_+$ is defined as 0 when the argument is less than 0 else it takes the value of the argument itself, $\mathbf{v} = \left(\sum_i \mathbf{C}_i^{-1}\right)^{-1} \sum_i \mu_i \mathbf{C}_i^{-1}$ and j is the index over the dimensions. The variance of the posterior is given by the Hessian which is defined as -

$$\tilde{\mathbf{S}} = \begin{cases} \left(\sum_i \mathbf{C}_i^{-1}\right)^{-1} & \tilde{\mu} \neq 0 \\ undefined & otherwise \end{cases} \tag{5.9}$$

When any component of the vector $\tilde{\mu}$ goes to zero we prune that particular dimension from the training data and from the model. This makes the learning very efficient.

To demonstrate the effectiveness of the sparsity prior in determining irrelevant variables, we use a 5 dimensional *sinc* dataset with 4 of its dimensions being irrelevant. We run the RVC algorithm with an ordinary Gaussian prior and compare it with a Laplacian ARD prior with α set to a value of 0.1. It must be noted that there exist a number of methods to learn the value of α (Figueiredo, 2003), but here we have used the value of α that gave the maximum value for the training likelihood with multiple restarts. The evolution of the test error with each epoch of the EM is shown in Fig. 5.7. We find that the algorithm with the ARD prior is able to converge faster by pruning the irrelevant dimensions.

In the final experiment we demonstrate the ability of RVC with an ARD prior to learn in high dimensions with many irrelevant input dimensions in the data. RVC is

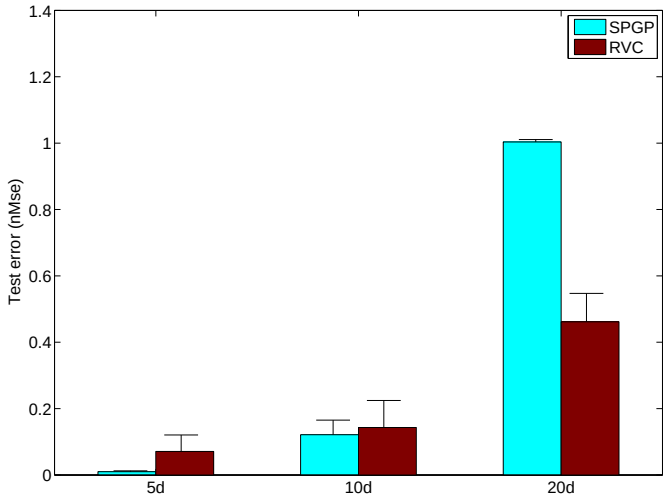


Figure 5.8: Performance of SPGP and RVC on datasets of increasing dimensions. Shown on the plot are the mean and a single standard deviation of 10 train-test splits of data with 500 train and test points.

run on a *sinc* dataset with 5, 10 and 20 dimensions out of which only one dimension is relevant. The test error of RVC is compared against SPGP in Fig. 5.8. The SPGP algorithm was run using the matlab implementation of sparse GP with default settings and using as many local models as the RVC. The result demonstrates that RVC is able to match up with SPGP in terms of its generalization ability. It must be noted that SPGP did not use an ARD prior since the results obtained for the ARD priors were worse than that of an ordinary RBF prior and requires careful initialization of parameters before training to obtain better generalization performance (Snelson & Ghahramani, 2006).

5.5 Discussion

In the initial part of this chapter we modified the training updates of RVC to enable learning from data in an online manner. This process was simplified due to the use of Bayesian inference procedure that naturally yields online updates.

In the second half of the chapter we provided extensive empirical evaluations to highlight the different strengths of the RVC formulation -

Generalization ability : Generalization ability of RVC was compared against other learning algorithms on well known multivariate benchmark datasets in Fig. 5.1 and was found to be fairly competitive compared to the other learning algorithms.

Online learning dynamics : RVC was shown to have a favourable online learning and fairly fast convergence as compared to related learning algorithms in Fig. 5.2.

Efficiency : RVC has a very efficient learning and prediction routines as compared to learning algorithms like GP and was demonstrated in Fig. 5.3.

Heteroscedastic noise : The ability to model heteroscedastic noise is another of the positives of RVC and is shown in Fig. 5.4.

Scalability : The RVC model scales up well with increasing number of data points learning them online. This is illustrated in experiments with the robot arm shown in Fig. 5.6 and it also scales up well with increasing number of input dimensions as shown in Fig. 5.8.

From these evaluations it can be seen that while RVC is well suited for a high dimensional data rich environment which requires fast learning with limited space complexity.

Chapter 6

Classification using Randomly Varying Coefficient model

In this chapter we look at a different scenario of learning where we are concerned with classifying given data points into different classes. The essential difference between a regression as we have been examining in the previous chapters, and classification lies in the nature of the response variable. In regression the response is continuous valued while in classification the response has discrete values corresponding to the discrete classes. While in regression we aim to approximate the mapping between the input and the response, in classification the aim is to *discriminate* between the classes conditional on the input. Research in classification has been dominated by kernel based methods like Support Vector Machine (SVM)(Vapnik, 1998) and more recently by non-parametric methods like Gaussian Process Classification (GP)(Rasmussen & Williams, 2006). Non-parametric methods like GP derives its success by using a covariance function of the input to model the dependency amongst the responses. The response for a test input is then computed as a linear smooth of all the training responses. This in turn leads to a large overhead in the time and space complexities for training and prediction. The training time complexity for GP is $O(N^3)$ for a data of size N and $O(N^2)$ for prediction. For an SVM, the training time complexity for an *active set* of size M in the worst case is $O(MN^2)$. These are significant overheads for large datasets. There have been many sparse Gaussian Process formulations (Csatò & Opper, 2002; Williams & Seeger, 2001; Smola & Bartlett, 2001; Tresp, 2000) that try to reduce this overhead but the efficiency still is quadratic in the size of the active set for most of these implementations. When we deal with *online* classifiers the space complexity also becomes an overarching concern. As seen in previous chapters Ran-

domly Varying Coefficient (RVC) model exhibits a good generalization performance at a low cost - $O(MN)$. Hence, we can use RVC to learn decision boundaries that would discriminate between classes.

6.1 Local logistic regression

We modify the RVC designed originally for regression into a classifier by using a logistic link function to convert a continuous response produced by a regressor into a value that lies between 0 and 1. This value in turn can be interpreted as the probability of the data belonging to a particular class. For a binary class variable z_i, the probability that $z_i = 1$ is modeled as the output of a logistic link function over a continuous latent variable y_i expressed as :

$$p(z_i = 1|y_i) = 1/(1+exp(-y_i))$$

In turn, the latent variable y_i is modelled as the response of a locally linear regression. For a locally linear region centered around $\mathbf{x}_c$ a conditional model for the continuous latent variable y_i is modelled as in eq. (3.1) and can be written as:

$$y_i = \beta_i^T \mathbf{x}_i + \varepsilon \tag{6.1}$$

where $\mathbf{x}_i \equiv [(\mathbf{x}_i' - \mathbf{x}_c)^T, 1]^T$ represents the center subtracted, bias augmented input vector, $\beta_i \equiv [\beta_i^{(1)} \dots \beta_i^{(d+1)}]^T$ represents the corresponding regression coefficient and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is the Gaussian mean zero noise with a variance σ^2. We follow the same formulation of RVC given in eq. (3.2) and define β_i as :

$$\beta_i \sim \mathcal{N}(\hat{\beta}, \mathbf{C}_i) \tag{6.2}$$

with $\mathbf{C}_i$ being a diagonal matrix given by :

$$\mathbf{C}_i(j,j) = \mathbf{x}_i^T \mathbf{x}_i / h_j^2$$

where h_j is the bandwidth of the jth dimension.

By marginalizing the hidden variables β_i we end up with a model for the local linear regression as :

$$y_i \sim \mathcal{N}(\hat{\beta}^T \mathbf{x}_i, \mathbf{x}_i^T \mathbf{C}_i \mathbf{x}_i + \sigma^2) \tag{6.3}$$

Proceeding along the same lines as Chapter 3 of (Rasmussen & Williams, 2006) and Section 5.3 of (Gelman & Hill, 2007) we now assume a noise-free latent variable y_i by setting σ^2 to zero. Setting σ^2 to zero yields the following model for y_i :

$$y_i \sim \mathcal{N}(\beta_i^T \mathbf{x}_i, 0)$$

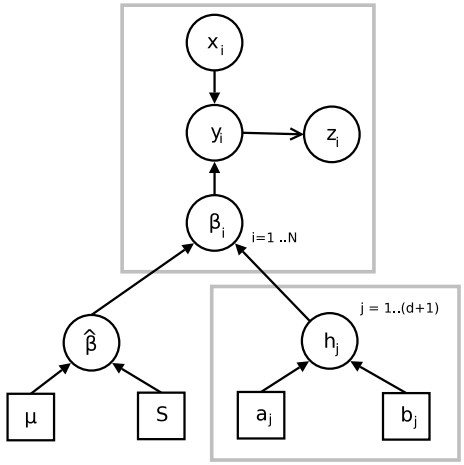


Figure 6.1: Probabilistic model of the logistic regressor

or equivalently marginalizing y_i :

$$p(z_i = 1|\beta_i) = 1/(1 + exp(-\beta_i^T \mathbf{x}_i)) \tag{6.4}$$

which corresponds to the classical formulation of a linear logistic regression with regression coefficients β_i.

We preserve the same probabilistic model as the original RVC for the rest of parameters. This includes a Gamma *regularizer* prior over the bandwidth parameters :

$$h_j^2 \sim Gamma(a_j, b_j) \tag{6.5}$$

and a noninformative Normal prior $\mathcal{N}(\mu, \mathbf{S})$ for the parameter $\hat{\beta}$. We assume a uniform prior for the regularizer hyperparameters a_j and b_j. The resultant probabilistic model for a *single* local model of the logistic regressor is shown in Fig. 6.1.

Next, we infer the posterior distribution over the parameters of the model using Bayesian inference rules.

6.2 Learning the parameters

The joint posterior over the parameters $\hat{\beta}$, h_j is given by:

$$P(\mathbf{h}, \hat{\beta}|\mathbf{z}, \mathbf{a}, \mathbf{b}, \mu, \mathbf{S}) = \frac{P(\mathbf{z}, \hat{\beta}, \mathbf{h}, \mathbf{a}, \mathbf{b}, \mu, \mathbf{S})}{P(\mathbf{z}, \mathbf{a}, \mathbf{b}, \mu, \mathbf{S})} \tag{6.6}$$

where we have used $\mathbf{h}$ to denote the vector $[h_1^2 \dots h_{d+1}^2]^T$ and $\mathbf{z}$ denotes the training data $[z_1 \dots z_N]^T$, $\mathbf{a} \equiv [a_1 \dots a_{d+1}]^T$ and $\mathbf{b} \equiv [b_1 \dots b_{d+1}]^T$. However, the posterior over the

parameters is rendered intractable due to the difficulty in evaluating the denominator of eq. (6.6). This necessitates the use of variational Bayesian EM to evaluate the posterior $P(\mathbf{h},\hat{\beta}|\mathbf{z},\mathbf{a},\mathbf{b},\mu,\mathbf{S})$ similar to Chapter 4 and learn the regularizer hyperparameters $\mathbf{a}$ and $\mathbf{b}$.

To learn the parameters of the model we can maximize the marginal log likelihood with respect to the parameters treating β_i as hidden variables. The marginal log likelihood is given by:

$$\begin{aligned}\mathcal{L} &= \ln P(\mathbf{z}|\mathbf{a},\mathbf{b},\mu,\mathbf{S}) \\ &= \ln \int P(\mathbf{z},\beta_1\ldots\beta_N,\mathbf{h},\hat{\beta}|\mathbf{a},\mathbf{b},\mu,\mathbf{S})d\beta_1\ldots d\beta_N d\mathbf{h} d\hat{\beta} \\ &= \ln \int \prod_i P(z_i|\beta_i)P(\beta_i|\hat{\beta},h_1,\ldots h_{d+1})\prod_j P(h_j^2|a_j,b_j)P(\hat{\beta}|\mu,\mathbf{S})d\beta_1\ldots d\beta_N dh_1\ldots dh_{d+1}d\hat{\beta}\end{aligned}$$

Using Jensen's inequality, the objective function that lower bounds $\mathcal{L}$ for a distribution Q is given by:

$$\mathcal{F} = \int Q(\beta_1\ldots\beta_N,\mathbf{h},\hat{\beta})\ln\frac{P(\mathbf{z},\beta_1\ldots\beta_N,\mathbf{h},\hat{\beta}|\mathbf{a},\mathbf{b},\mu,\mathbf{S},c,d)}{Q(\beta_1\ldots\beta_N,\mathbf{h},\hat{\beta})}d\beta_1\ldots d\beta_N d\mathbf{h} d\hat{\beta} \quad (6.7)$$

The optimal value for $Q(\beta_1\ldots\beta_N,\mathbf{h},\hat{\beta})$ that makes the bound tight is given by the joint posterior $P(\beta_1\ldots\beta_N,\mathbf{h},\hat{\beta}|\mathbf{z})$ but since this posterior is intractable, we make a factorized approximation by assuming that the posterior over the variables is independent and can be expressed as :

$$Q(\beta_1\ldots\beta_N,\mathbf{h},\hat{\beta}) = \prod_i Q(\beta_i|\mathbf{z})\prod_j Q(h_j^2|\mathbf{z})Q(\hat{\beta}|\mathbf{z})$$

Substituting the factorized approximation in eq. (6.7) we get:

$$\begin{aligned}\mathcal{F}_{approx} = &\sum_i \langle \ln P(z_i|\beta_i)\rangle_{Q_{\beta_i}} + \left\langle \ln P(\beta_i|\hat{\beta},h_1\ldots h_{d+1})\right\rangle_{Q_{\beta_i},Q_{h_1}\ldots Q_{h_{d+1}},Q_{\hat{\beta}}} \\ &+\sum_j \left\langle \ln P(h_j^2|a_j,b_j)\right\rangle_{Q_{h_j}} + \left\langle \ln P(\hat{\beta}|\mu,\mathbf{S})\right\rangle_{Q_{\hat{\beta}}} \\ &-\sum_i \left\langle \ln Q_{\beta_i}\right\rangle_{Q_{\beta_i}} - \sum_j \left\langle \ln Q_{h_j}\right\rangle_{Q_{h_j}} - \left\langle \ln Q_{\hat{\beta}}\right\rangle_{Q_{\hat{\beta}}}\end{aligned} \quad (6.8)$$

where $\langle . \rangle_Q$ denotes the expectation with respect to the distribution Q. The optimal values of the posterior probabilities can be computed iteratively by maximizing the functional $\mathcal{F}_{approx}$ with respect to each individual posterior distribution keeping the other distributions fixed akin to EM. For a logistic regression, an additional complication arises in the computation of the posterior distribution over the hidden variables

β_i. The likelihood term $P(z_i|\beta_i)$ is given by the logistic link function whereas the prior over β_i is Gaussian and is not conjugate to the likelihood term. We solve this issue by using a Laplacian approximation to approximate the posterior over β_i by a Gaussian distribution.

6.2.1 Laplace approximation of $Q(\beta_i|\mathbf{z})$

Frequently in Bayesian inference of complicated probabilistic models that employs non-conjugate priors, the posterior is approximated by a Gaussian distribution centered around the mode of the actual posterior. This is usually termed as the Laplace approximation (Tierney & Kadane, 1986). In our case the log posterior over the hidden variable β_i is given by :

$$\begin{aligned}\mathcal{M} = \ln Q(\beta_i|\mathbf{z}) = \ln P(z_i|\beta_i) + \left\langle \ln P(\beta_i|\hat{\beta}, \mathbf{C}_i) \right\rangle_{Q(\hat{\beta}),Q(\mathbf{h})} \\ - \ln \int exp(\ln P(z_i|\beta_i) + \left\langle \ln P(\beta_i|\hat{\beta}, \mathbf{C}_i) \right\rangle_{Q(\hat{\beta}),Q(\mathbf{h})})\end{aligned} \tag{6.9}$$

Laplace approximation of the posterior corresponds to

$$Q(\beta_i|\mathbf{z}) \sim \mathcal{N}(\nu_i, \mathbf{G}_i)$$

where $\nu_i = argmax_{\beta_i}\mathcal{M}$ and $\mathbf{G}_i^{-1} = -\nabla\nabla\mathcal{M}|_{\beta_i=\nu_i}$ is the Hessian of the negative log posterior. The posterior mode ν_i can be obtained by setting the gradient of $\mathcal{M}$ to zero. However, this procedure does not yield a closed form solution for the posterior mode ν_i. We then have to resort to Newton's update to find the mode iteratively as shown :

$$\nu_i = \nu_i^{old} - (\nabla\nabla\mathcal{M})^{-1}\nabla\mathcal{M} \tag{6.10}$$

Substituting the forms of $P(\beta_i|\hat{\beta}, \mathbf{C}_i)$ and $P(z_i|\beta_i)$ from eqs. (6.2) and (6.4) into eq. (6.9) and differentiating it with respect to β_i we get :

$$\nabla\mathcal{M}\ |_{\beta_i=\nu_i^{old}} = \mathbf{x}_i(z_i - \pi_i) - \langle\mathbf{C}_i\rangle^{-1}(\nu_i^{old} - \tilde{\mu}) \tag{6.11}$$

$$\nabla\nabla\mathcal{M}\ |_{\beta_i=\nu_i^{old}} = -\mathbf{x}_i\mathbf{x}_i^T\pi_i(1-\pi_i) - \langle\mathbf{C}_i\rangle^{-1} \tag{6.12}$$

where $\pi_i = 1/(1 + exp(-\mathbf{x}_i^T\nu_i^{old}))$ and $\langle\mathbf{C}_i\rangle = diag(1/\left\langle h_j^2 \right\rangle_{Q(h_j^2)})$. It can be found from eq. (6.12) that $\mathbf{G}_i^{-1} = -\nabla\nabla\mathcal{M} = \mathbf{x}_i\mathbf{x}_i^T\pi_i(1-\pi_i) + \langle\mathbf{C}_i\rangle^{-1}$ and the estimate for ν_i can be obtained by substituting eqs. (6.11) and (6.12) in eq. (6.10) yielding :

$$\nu_i = \nu_i^{old} + \mathbf{G}_i(\mathbf{x}_i(z_i - \pi_i) - \langle\mathbf{C}_i\rangle^{-1}(\nu_i^{old} - \tilde{\mu}))$$

which can be simplified using Sherman-Morrison Woodbury theorem to yield :

$$\mathbf{G}_i = \langle \mathbf{C}_i \rangle - \frac{\langle \mathbf{C}_i \rangle \mathbf{x}_i \mathbf{x}_i^T \langle \mathbf{C}_i \rangle}{w_i + \mathbf{x}_i^T \langle \mathbf{C}_i \rangle \mathbf{x}_i} \tag{6.13}$$

$$\nu_i = \frac{\langle \mathbf{C}_i \rangle \mathbf{x}_i}{(w_i + \mathbf{x}_i^T \langle \mathbf{C}_i \rangle \mathbf{x}_i)} ((z_i - \pi_i) w_i + \mathbf{x}_i^T \nu_i^{old} - \mathbf{x}_i^T \tilde{\mu}) + \tilde{\mu} \tag{6.14}$$

where $w_i = \frac{1}{\pi_i(1-\pi_i)}$.

6.2.2 Posteriors for $Q(\hat{\beta}|\mathbf{z})$ and $Q(h_j^2|\mathbf{z})$

Posterior over $\hat{\beta}$ based on the likelihood and the prior over $\hat{\beta}$ can be derived as :

$$Q(\hat{\beta}|\mathbf{z}) \sim \mathcal{N}(\tilde{\mu}, \tilde{\mathbf{S}})$$

where

$$\tilde{\mathbf{S}} = (\sum_i \langle \mathbf{C}_i \rangle^{-1} + \mathbf{S}^{-1})^{-1} \tag{6.15}$$

$$\tilde{\mu} = \tilde{\mathbf{S}} (\sum_i \langle \mathbf{C}_i \rangle^{-1} \nu_i + \mathbf{S}^{-1} \mu) \tag{6.16}$$

Similarly, the posterior over h_j is given by :

$$Q(h_j^2|\mathbf{z}) \sim Gamma(\tilde{a}_j, \tilde{b}_j)$$

where

$$\tilde{a}_j = a_j + N/2 \tag{6.17}$$

$$\tilde{b}_j = b_j + \sum_i \left[(\nu_{i,j} - \tilde{\mu}_{i,j})^2 + \mathbf{G}_{i,jj} + \tilde{\mathbf{S}}_{jj} \right] / 2\mathbf{x}_i^T \mathbf{x}_i \tag{6.18}$$

where $\nu_{i,j}$ and $\tilde{\mu}_{i,j}$ represent the j^{th} element of the vectors. The regularizer hyperparameters a_j and b_j in eqs. (6.17) and (6.18) are obtained by maximizing the bound $\mathcal{F}_{approx}$ given by eq. (6.8) with respect to these hyperparameters keeping the posterior distributions Q fixed. This leads to the same update rule as eq. (4.14) for the hyperparameters and is given by:

$$a_j = \tilde{a}_j, \quad b_j = \tilde{b}_j \tag{6.19}$$

The posterior parameters are inferred by using a partial Newton step to infer the posterior of β_i followed by EM updates. The procedure for training a local model is summarised in Algorithm 5.

Algorithm 5 Training a local model

1: Initialize hyperparameters: $\Theta \equiv \{\mu_0, \mathbf{S}, \mathbf{a}, \mathbf{b}\}$.
2: Input: Batch training data $\mathbf{X}$, $\mathbf{z}$
3: **repeat**
4: Initialize $\nu_i^{old} = \tilde{\mu}$, $\pi_i = 1/(1+exp(-\mathbf{x}_i^T \nu_i^{old}))$ and $w_i = \frac{1}{\pi_i(1-\pi_i)}$.
5: Estimate posterior hyperparameters $\tilde{\Theta}$ using Θ and eq. (6.13), (6.14) and eqs. (6.15) - (6.18).
6: Estimate values of the hyperparameters $\mathbf{a}$ and $\mathbf{b}$ of the regularizer prior using eq. (6.19).
7: **until** convergence of $\tilde{\Theta}$

6.3 Prediction

Using the learning procedure discussed in the previous section we obtain independently trained local models of the logistic regression. Each of the local models represent a separate classifier with a linear decision boundary. To obtain an aggregate prediction for a particular query input we need to combine the classifiers.

Ensemble learning has been a field of research which has seen considerable amount of research into the ways of combining classifiers (Kittler et al., 1998). In this thesis though we use the same technique as RVC - combining the linear regressors to produce a non-linear regression model and then using a logistic transform to obtain a classifier.

Given the ensemble of trained local experts, in order to predict the response y_q for a new query point $\mathbf{x}_q$, we take the normalized product of the *predictive distribution* of each local expert. This results in the predictive distribution for the k-th local model :

$$y_{q,k} \sim \mathcal{N}(\tilde{\mu}^T \mathbf{x}_{q,k}, \mathbf{x}_{q,k}{}^T (\tilde{\mathbf{S}}_k + \mathbf{C}_{k\mathbf{h}_{mode}}) \mathbf{x}_{q,k})$$

where $\mathbf{x}_{q,k}$ refers to the query point with the k-th center subtracted and augmented with bias. Blending the prediction of different experts by taking their product and normalizing it results in a Normal distribution given by:

$$y_q \sim \mathcal{N}(\mu, \zeta^2) \quad \text{where} \quad \mu = \frac{\sum_k \alpha_k \tilde{\mu}_k^T \mathbf{x}_{q,k}}{\sum_k \alpha_k}, \;\; \zeta^2 = \frac{1}{\sum_k \alpha_k}. \tag{6.20}$$

Here, μ is a sum of the means of each individual expert weighted by the confidence expressed by each expert in its own prediction α_k, ζ^2 is the variance and α_k is the precision of each expert:

$$\alpha_k = 1/(\mathbf{x}_{q,k}^T (\tilde{\mathbf{S}}_k + \mathbf{C}_k) \mathbf{x}_{q,k}) \tag{6.21}$$

Algorithm 6 Global prediction using local models

Input: Query point $\mathbf{x}_q$
Initialize: $sum_\alpha = 0$, $y_q = 0$
for k = 1 to #local models **do**
    $\mathbf{x}_{q,k} = \mathbf{x}_q - \mathbf{x}_{c,k}$
    Calculate α_k using eq. (6.21)
    $y_q = y_q + \alpha_k \tilde{\mu}_k^T \mathbf{x}_{q,k}$
    $sum_\alpha = sum_\alpha + \alpha_k$
end for
$y_q = y_q / sum_\alpha$
Output : $P(z_q = 1) = 1/(1 + exp(-y_q))$

The predictive probability for the logistic regression can be obtained by combining the predictive probability of the latent variable y_q with the link function and marginalizing the latent variable to yield :

$$P(z_q = 1|\mathbf{z}) = \int P(z_q = 1|y_q) P(y_q|\mathbf{z}) dy_q \tag{6.22}$$

where $P(z_q = 1|y_q)$ is a logistic function and $P(y_q|\mathbf{z})$ is the predictive distribution given by eq. (6.20). The integral given in eq. (6.22) cannot be evaluated analytically and we must rely on numerical methods or sampling to evaluate the integral. In the context of binary classification if we threshold the predictive probability at $\frac{1}{2}$ in order to discriminate between classes a MAP prediction would be the same as an averaged prediction as shown in (Bishop, 1995) and explained in (Rasmussen & Williams, 2006). Therefore we use MAP predictive estimate for classification. To obtain the MAP prediction we evaluate the integral in eq. (6.22) by approximating $P(y_q|\mathbf{z})$ by a delta function at its mode. The prediction routine is listed in Algorithm 6.

6.4 Evaluation

In the first evaluation, we look at the ability of logistic RVC (lRVC) to discriminate between two classes on a one dimensional artificial data. The data used in the task has previously been used in (Rasmussen & Williams, 2006) and is shown in Fig. 6.2. The data consists of two classes distributed such that there is linear separability at one region (between cluster centered at -6 and cluster centered at 0) and there is no clear separation in the other region (cluster centered at 0 and cluster centered at 2). This

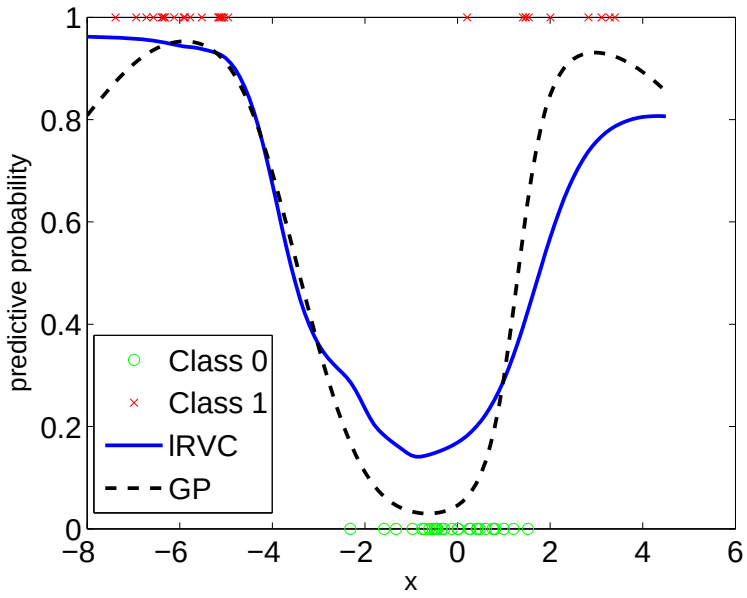


Figure 6.2: Comparison of the decision boundary learnt by lRVC and GP

data was used to train lRVC with local models placed at all the training points and a Laplacian GP classifier with a square exponential covariance function. The predictive probability curve learnt by the two classifiers is compared in Fig. 6.2. It is interesting to observe that the predictive probabilities assigned by lRVC to the data points lying in the inseparable region is less than the probabilities assigned to the separable part of the class. The local modeling of lRVC ensures that the decision boundaries learnt by the local models is based on the local spatial distribution of the class and hence in this case is more confident about the linearly separable part of the input space than the non-separable data. On the other hand, GP assigns almost the same and sometimes more (Rasmussen & Williams, 2006) predictive probability to the inseparable points than the separable ones which is not very intuitive for a classifier.

Before we proceed to detailed evaluation experiments, we need to specify the evaluation measures that would be used to compare different classifiers. We compare classifiers based on two different measures - *misclassification error* and *target information*. The former is the often used loss function that measures the mean number of misclassifications produced by a classifier on a test set. The *target information* criteria refers to a loss function that takes into account the confidence expressed by the classifier about

its prediction. The loss function is given by :

$$I = \frac{1}{N}\left[\sum_{z_i=1} log_2(P(z_i = 1|x_i^q)) + \sum_{z_i=0} log_2(1 - P(z_i = 1|x_i^q))\right] + 1$$

and it measures in bits, the information conveyed by the classifier about the test target. For a baseline classifier that assigns classes at random $I \rightarrow 0$ and for a more confident discrimination of classes $I \rightarrow 1$.

6.4.1 Comparison of generalization performance and time efficiency

In the next evaluation, we compare the generalization performance of lRVC against a Gaussian Process classifier and a baseline probabilistic linear logistic regressor. The lRVC used in the evaluation used around 20 local models initialized at the cluster centers in the input space. The Gaussian Process uses a square exponential kernel and a logistic link function. The three classifiers are compared on different benchmark datasets listed in Table 6.1. The Breast cancer, Heart (Cleveland) and the Ionosphere dataset were obtained from the UCI repository, Pima and synthetic datasets are the same as the ones used in (Ripley, 1996) [1]. The USPS dataset corresponds to the digit discrimination task listed in (Rasmussen & Williams, 2006). The Catalysis and Gatineau datasets were obtained from the predictive uncertainty challenge [2] were the validation set has been used as test set. The evaluations on the datasets obtained from UCI was carried out on 10 train-test splits of the data and the mean and standard deviations are reported here. For all other datasets a single train-test trial was carried out using the train and test files provided. This makes it possible to compare other classifiers that have previously used the latter datasets. For the Gatineau dataset GP was trained using a subset of 1000 training points due to practical considerations of time and space complexity. The evaluation statistics are listed in Table 6.2. Also shown in the table is the results for a LIBSVM(Chang & Lin, 2001) (with an RBF kernel) evaluation over the same datasets. The comparison for SVM is restricted to the misclassification error since SVM does not provide a predictive probability. One can see from the results that lRVC is able to match the performance of GP for all the datasets and outperforms the baseline linear classifier especially when the target information is used for the comparison. It must be noted that while lRVC used only a small number of local models for prediction, GP used all of the training set for training and prediction. To emphasize this difference Table 6.3 shows the time taken by lRVC and GP for training and prediction on a dataset

[1] The datasets can be obtained from http://www.stats.ox.ac.uk/pub/PRNN/

[2] http://predict.kyb.tuebingen.mpg.de/pages/home.php

Dataset	#train pts.	#test pts	#dim
Breast cancer	142	427	30
Heart(Cleveland)	149	148	13
Ionosphere	175	176	33
Pima	200	332	7
Synthetic	250	1000	2
USPS(3-5)	767	773	256
Catalysis	873	300	617
Gatineau	3000	2176	1092

Table 6.1: Statistics of the benchmark datasets

	lRVC		GP		Linear		SVM
	Error	Information	Error	Information	Error	Information	Error
Breast	0.028(0.007)	0.807(0.010)	0.026(0.009)	0.805(0.045)	0.042(0.014)	0.797(0.050)	0.028(0.009)
Heart	0.166(0.017)	0.432(0.049)	0.169(0.017)	0.423(0.039)	0.173(0.024)	0.388(0.113)	0.173(0.022)
Ionosphere	0.152(0.027)	0.338(0.170)	0.123(0.025)	0.535(0.054)	0.163(0.027)	-2.288(0.963)	0.078(0.025)
Pima	0.202	0.361	0.222	0.276	0.198	0.364	0.198
Synthetic	0.100	0.649	0.093	0.658	0.114	0.611	0.100
USPS(3-5)	0.045	0.798	0.025	0.794	0.040	0.476	0.023
Catalysis	0.303	0.143	0.303	0.150	0.343	-3.516	0.323
Gatineau	0.090	0.570	0.090	0.588	0.154	-0.484	0.090

Table 6.2: Performance comparison between lRVC and GP in terms of the misclassification error rate and the target information (measured in bits) conveyed by the classifier. The values in parenthesis indicate the standard deviation.

consisting of the USPS digit 3 classified against the rest of the digits. lRVC can be seen to achieve a good generalization performance with a low overhead.

6.4.2 Rejection using the predictive confidence bounds

In the last evaluation, we evaluate the confidence bounds learnt by lRVC by plotting the relation between the reject rate, the misclassification error and the target information. In this experiment the first and second moments for the predictive probability were computed by using sampling to evaluate the integral in eq. (6.22). The test samples which had a variance above a threshold were rejected and the misclassification error was evaluated for the rest of the test data. The dataset used for this purpose was the

Method		0	1	2	4	5	6	7	8	9
GP	Error	0.0108	0.0028	0.0183	0.0046	0.0259	0	0.0024	0.0254	0.0061
	Information	0.8944	0.9027	0.8627	0.8686	0.7943	0.8857	0.8677	0.8550	0.8486
	Train time(sec)	1915.9	1475.8	1020.8	877.0	911.4	963.7	919.9	845.8	993.0
	Test time(sec)	47.8	37.6	25.0	22.1	20.2	23.8	23.4	20.5	24.4
lRVC	Error	0.009	0.005	0.022	0.005	0.045	0.003	0.009	0.032	0.012
	Information	0.944	0.960	0.886	0.966	0.798	0.972	0.955	0.883	0.946
	Train time(sec)	582.59	509.67	440.44	402.23	382.08	400.75	370.25	363.13	383.42
	Test time(sec)	0.87	0.74	0.61	0.61	0.55	0.55	0.54	0.49	0.55

Table 6.3: Comparison between the time taken for training and prediction by lRVC and GP on the USPS dataset.

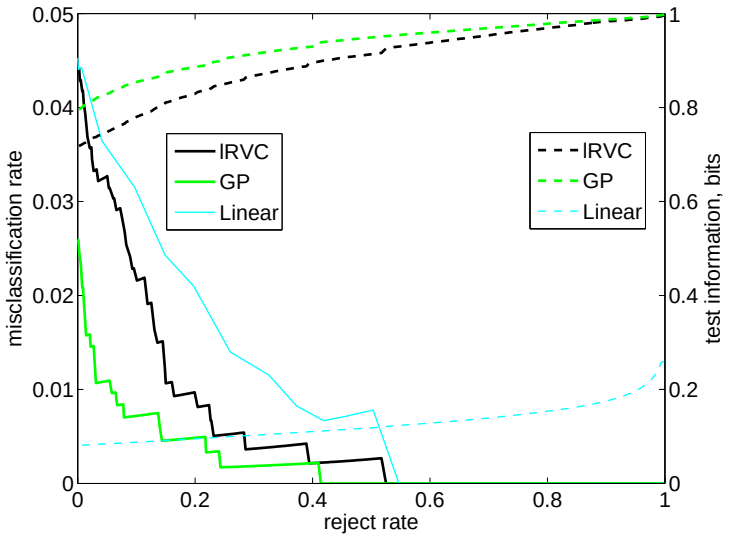


Figure 6.3: Comparison of error-reject curve for lRVC and GP

USPS data. The misclassification error typically decreases as test samples are rejected and an ideal classifier would have a larger reduction in the misclassification error with respect to the rejection rate. The error and the target information versus the rejection rate for lRVC, GP and the Bayesian linear logistic regressor were evaluated and plotted in Fig. 6.3. It can be seen that lRVC's performance exceeds that of the linear classifier by a large margin and is not significantly different from GP.

6.4.3 Dynamics of online learning

Similar to Chapter 5 we can derive the online updates for the batch updates. The online updates for the logistic RVC are the same as eqs. (5.1) - (5.4) derived in Chapter

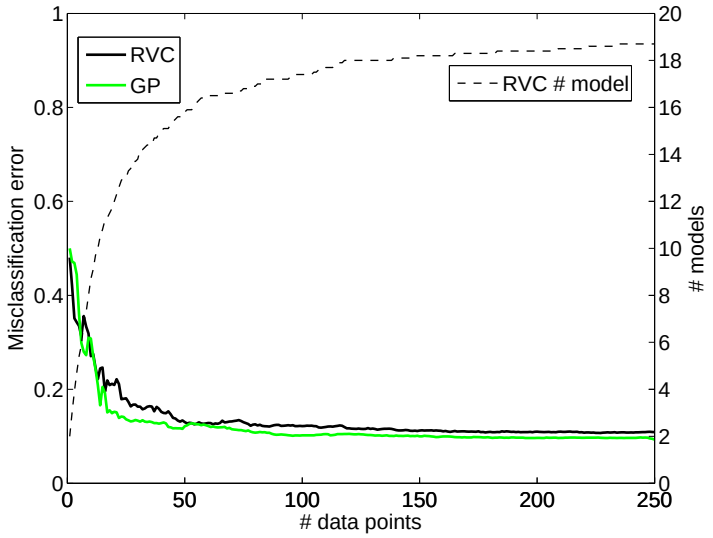


Figure 6.4: Online learning dynamics of lRVC compared with GP. The plots are the average performance over 10 trials of different orders of data presentations.

5. In the last evaluation, we use these online updates to learn a classifier on the *synthetic* dataset. The data points are presented to the online learner one at a time and the misclassification error is evaluated over the test data after each training update. The dynamics of the learning process is shown in Fig. 6.4. The learning dynamics is compared with the generalization performance of GP which uses increasing number of training data points and the corresponding test error at each stage is displayed. It can be seen from the figure that online version of lRVC exhibits fast convergence and matches the performance of GP asymptotically.

6.5 Discussion

In this chapter we extended the learning framework of RVC from a regression setting to handle classification. We used a logistic link function to turn regression into classification. From the evaluations in this chapter we can see that local logistic regression is a very competitive method. The result makes it more significant when we take it into account that the logistic regression is able to achieve such a good performance using a small number of local models. Moreover the time and space efficiency is linear in terms of the data points and the dimension. In contrast, kernel classification paradigms like GP and SVM have a much higher overhead in training and testing.

The use of variational Bayesian EM approximation for learning the parameters of the model allows us to reformulate the learning rules into a set of online updates for the parameters that would enable the logistic classifier to learn from data in real time.

The logistic regression formulation in this thesis is restricted to binary classification. It can be easily extended to a multi-class classification using a softmax link function instead of a logistic link function. The treatment of the learning remains the same in that case too.

In conclusion, the contribution of this chapter has been a probabilistic formulation of a local linear logistic regressor that can learn online using very efficient Bayesian updates and at the same time exhibits good learning characteristics.

Chapter 7

Contributions and future work

The major contribution of the thesis is the development of a Bayesian formulation for independent spatially localised learners for multivariate nonlinear regression. We have used a novel formulation of data dependent priors in order to carve out locally linear regions while avoiding competition amongst local models. The 'non-competitive' behavior of each local model allows independent, efficient learning while the Bayesian regularizer hyperpriors guard against the danger of overfitting or over-smoothing through automatic local bandwidth adaptation. In this chapter we summarize the contributions made in this thesis along with future avenues of research possible.

In this thesis, use of a product of regression experts explained in Chapter 2 has been shown to be a feasible approach to follow when we require efficient learners that can learn in an online manner and dynamically adjust model complexity. Furthermore in this thesis we have derived the connection between the ambiguity formulation of regression ensembles and a product of regression experts in eq. (2.5). Another novelty is the use of POE to bridge the gap between previously unrelated pieces of research - conditional random fields, independent ensemble learning and complementary prior. Though it is difficult to assimilate independent ensemble learning into a probabilistic framework, it is useful in designing efficient learning systems. Therefore it is worthwhile to compare the framework of dependent learning to that of independent learning by treating the latter as an approximation to the dependent learning. In eq. (2.10) we provide a bound that characterizes the error between an independent and a dependent learning. In future this bound could be further refined to quantify the effect of independent learning on the generalization ability of a learning algorithm.

Chapter 3 provides a unique probabilistic formulation of a locally weighted learning using a heteroscedastic variance component to weight the errors. This type of a

model has not been used for locally weighted learning, nonetheless, eq. (3.8) demonstrates the equivalence of heteroscedastic model to a locally weighted regression. Section 3.3.1 examines the ability of RVC to adapt its model complexity and illustrates the change in the complexity of the learning model through a plot of the varying degrees of freedom in Fig. 3.6. The plot provided in Fig. 3.6 is a result of an empirical analysis of the degrees of freedom; it might be possible to extend this to a more analytical treatment wherein we can derive a closed form functional relation between the degrees of freedom of RVC and the bandwidths of individual local models.

Chapter 4 provides a simple Variational Bayesian EM procedure to train the local models efficiently. These learning updates are very efficient with each EM update being $O(MN)$ where M is the number of local models and N the number of training points. Furthermore, Bayesian inference procedure also yields a predictive distribution with a mean prediction and a confidence bound on the prediction. The predictive distribution of RVC is shown to provide meaningful confidence bounds through an illustrative example in Fig. 4.2. The Variational Bayesian algorithm provides efficient updates but the approximation itself tends to have a slow convergence rate. One of the possible avenues of future research will be to use algorithms like (Qi & Jaakkola, 2006) to speed up the convergence of the VBEM algorithm.

In Chapter 5 we derived the online updates for the Variational Bayesian EM learning and evaluated the RVC model against the state of the art in non-linear regression techniques on artificial as well as real world data sets. RVC matched the generalization performance of LWPR while avoiding cumbersome parameter tuning for initialization (refer Fig. 5.1, Fig. 5.2). It achieves competitive performance compared to GP - essentially a batch method, while being much more computationally efficient (refer Fig. 5.3). One of the main advantages of spatially localised models of RVC is its ability to model locally constant but globally varying properties of a function like heteroscedastic noise. The ability of RVC to model heteroscedastic noise is illustrated in Fig. 5.4 where it is compared with a GP on a benchmark dataset and is shown to perform better than a GP. The RVC is also shown to scale well with increasing dimensions of the input space in Section 5.4 where an ARD prior is used to learn in a high dimensional space with lots of irrelevant dimensions and its performance is compared with other learning methods in Fig. 5.8. In Fig. 5.6 we also demonstrate a practical application of RVC in learning the inverse model of robotic arm and is shown to perform well compared to other similar methods. The pros and cons of RVC compared against other learning algorithms have been compared in Table 7.1. The space and computational efficiency of RVC

coupled with the ability to grow model complexity in a data driven fashion makes it a strong candidate for practical online and real time learning scenarios and was the basis for (Edakunni et al., 2007)

The probabilistic formulation of RVC can further be exploited to extend it to generalized linear models and has been illustrated in Chapter 6 by formulating a logistic regression based on the model of RVC. The classifier based on RVC was also evaluated against the state of the art methods and was found to exhibit good generalization properties with a low computational complexity as illustrated by Tables 6.2 and 6.3. This work formed the basis for (Edakunni & Vijayakumar, 2009).

Active learning is another area of research that can be pursued in the context of the work presented in this thesis. While traditional research has concentrated on active learning in global models of learning, the localised independent models of RVC provide an alternative strategy for active learning. The active learning itself can be localised with the learning of the fit and the number of models being learnt actively. This effort is aided by the presence of a probabilistic confidence bound provided by each of the local models constituting the ensemble.

To conclude, this thesis presents an online, spatially localised ensemble learning method with efficient Bayesian inference rules that provides competitive generalization results for a variety of learning scenarios.

Features	RVC	LWPR	GP	SPGP
Non-parametric	Yes	Yes	Yes	Yes
Online	Yes	Yes	No	No
IID assumption for data	Yes	Yes	No	No
Probabilistic framework	Yes	No	Yes	Yes
Training method	Variational Bayesian EM	Leave-one-out cross validation	MAP estimate of hyperparameters	MAP estimate of hyperparameters
Time complexity for training	$O(M)$ per data point	$O(M)$ per data point	$O(N^3)$ per gradient descent step	$O(M^2N)$ per gradient descent step
Time complexity for prediction	$O(M)$	$O(M)$	$O(N^2)$	$O(M)$
Space complexity	$O(M)$	$O(M)$	$O(N^2)$	$O(MN)$
Handling high dimensions	ARD prior	Projected regression	ARD prior	Scales poorly on ARD prior
Open parameters	All parameters other than bandwidth have non-informative prior	Number of open parameters	None	Few open parameters
Number of data points for convergence on training	Large	Large	Small	Small

Table 7.1: Comparison of the features of the learning algorithms discussed in this thesis. Note that here, M is the number of models and N the number of training data points.

Appendix A

Variational Bayesian Expectation Maximization

In this section we introduce the variational Bayesian expectation maximization algorithm as applied to a generic graphical model involving hidden variables and distributions over those variables. We start with a simple model with the visible data given by $\mathbf{D}$ which stands for the collection of training data of N points whose i^{th} individual point is represented by $\mathcal{D}_i$. The hidden variable associated with each individual data point is given by the variable z_i. The likelihood is given by $P(\mathbf{D}|z_1 \dots z_N)$. Assuming that the data is conditionally independent the likelihood can be written down as :

$$P(\mathbf{D}|z_1 \dots z_N) = \prod_{i=1}^{N} P(\mathcal{D}_i|z_i) \tag{A.1}$$

The hidden variables are assumed to have been generated from a distribution given by : $P(z_1 \dots z_N|\theta)$ where θ is the parameter of the prior distribution. The prior distribution over the hidden variables is assumed to be conditionally independent :

$$P(z_1 \dots z_N|\theta) = \prod_{i=N} P(z_i|\theta) \tag{A.2}$$

and the prior over the parameter θ is given by $P(\theta|\alpha)$ where α is the hyperparameter.

We are now interested in inferring the posterior distribution over the parameter θ. Following Bayes' rule, the posterior is given by :

$$P(\theta|\mathbf{D}) = \frac{P(\mathbf{D}|\theta)P(\theta|\alpha)}{P(\mathbf{D}|\alpha)} \tag{A.3}$$

Here, $P(\mathbf{D}|\theta)$ is obtained by integrating over the hidden variables from the joint distribution :

$$P(\mathbf{D}|\theta) = \int P(\mathbf{D}|z_1 \dots z_N)P(z_1 \dots z_N|\theta)dz_1 \dots dz_N \tag{A.4}$$

The integration in eq. (A.4) leads to a closed form expression for $P(\mathbf{D}|\theta)$ only when $P(z_1 \dots z_N|\theta)$ is conjugate to $P(\mathbf{D}|z_1 \dots z_N)$. We assume from hereon that the distribution is conjugate. The prior over the parameter $P(\theta|\alpha)$ is also assumed to be conjugate to $P(z_1 \dots z_N|\theta)$. Despite these pairwise conjugate distributions the posterior over the parameter θ is rendered intractable due to the difficulty in evaluating the evidence term $P(\mathbf{D}|\alpha)$ in the expression in eq. (A.3). The cause of this problem is the integration involved in evaluating the evidence term :

$$P(\mathbf{D}|\alpha) = \int P(\mathbf{D}|\theta)P(\theta|\alpha)d\theta$$

and this would lead to a closed form expression only when $P(\theta|\alpha)$ is conjugate to $P(\mathbf{D}|\theta)$ which is generally not the case.

In order to infer the posterior over the parameters we can start with the marginalized likelihood given by :

$$P(\mathbf{D},\alpha) = \int P(\mathbf{D}|\mathbf{z})P(\mathbf{z}|\theta)P(\theta|\alpha)d\mathbf{z}d\theta \tag{A.5}$$

where $\mathbf{z} \equiv [z_1 \dots z_N]$. Taking the log of the likelihood given in eq. (A.5) we get :

$$\mathcal{L} = \ln P(\mathbf{D}|\alpha) = \ln \int P(\mathbf{D}|\mathbf{z})P(\mathbf{z}|\theta)P(\theta|\alpha)d\mathbf{z}d\theta$$

Using Jensen's inequality $\mathcal{L}$ can be lower bounded by $\mathcal{F}$ given by :

$$\begin{aligned} \mathcal{L} &= \ln \int P(\mathbf{D}|\mathbf{z})P(\mathbf{z}|\theta)P(\theta|\alpha)d\mathbf{z}d\theta & \text{(A.6)} \\ &\geq \int Q(\mathbf{z},\theta) \ln \frac{P(\mathbf{D}|\mathbf{z})P(\mathbf{z}|\theta)P(\theta|\alpha)}{Q(\mathbf{z},\theta)} d\mathbf{z}d\theta & \text{(A.7)} \\ &= \mathcal{F} & \text{(A.8)} \end{aligned}$$

where $Q(\mathbf{z},\theta)$ is any arbitrary function of $\mathbf{z}$ and θ that integrates to unity. The inequality of eq. (A.7) becomes an equality when $Q(\mathbf{z},\theta)$ equals the joint posterior of the random variables. But as we had seen earlier this distribution is intractable and our aim is to estimate this posterior distribution.

We can constrain the form of the joint posterior as a factorized form of distribution which implies $Q(\mathbf{z},\theta) \approx Q(\mathbf{z})Q(\theta)$. Substituting this factorized form in eq. (A.7) we get :

$$\mathcal{F} \approx \mathcal{F}_{approx} = \int Q(\mathbf{z})Q(\theta) \ln \frac{P(\mathbf{D}|\mathbf{z})P(\mathbf{z}|\theta)P(\theta|\alpha)}{Q(\mathbf{z})Q(\theta)} d\mathbf{z}d\theta \tag{A.9}$$

Here, $\mathcal{F}_{approx}$ can be seen to be a functional form of the distributions $Q(\mathbf{z})$ and $Q(\theta)$ and we can maximize $\mathcal{F}_{approx}$ iteratively with respect to the free distributions. These iterative steps would essentially form the updates for the Variational Bayesian EM (VBEM) algorithm.

A.1 VBE step

In the VBE step we try to estimate the posterior distribution over the hidden variables $Q(\mathbf{z})$ and is obtained by taking the functional derivative of $\mathcal{F}_{approx}$ defined by eq. (A.9) with respect to $Q(\mathbf{z})$ and equating it to zero :

$$\begin{aligned}
\frac{\partial \mathcal{F}_{approx}}{\partial Q(\mathbf{z})} &= \int \left[\frac{\partial \int Q(\mathbf{z}) \ln P(\mathbf{D}, \mathbf{z}|\theta, \alpha) d\mathbf{z}}{\partial Q(\mathbf{z})} \right] Q(\theta) d\theta && \text{(A.10)} \\
&= \int \left[\ln P(\mathbf{D}, \mathbf{z}|\theta, \alpha) - \ln Q(\mathbf{z}) - 1 \right] Q(\theta) d\theta && \text{(A.11)} \\
&= 0 && \text{(A.12)}
\end{aligned}$$

Solving for $Q(\mathbf{z})$ we get :

$$\ln Q(\mathbf{z}) \propto \int \ln P(\mathbf{D}, \mathbf{z}|\theta, \alpha) Q(\theta) d\theta \tag{A.13}$$

Substituting the form of $P(\mathbf{D}, \mathbf{z}|\theta, \alpha)$ from eq. (A.1) and (A.2) we get :

$$\begin{aligned}
\ln Q(\mathbf{z}) &\propto \int \sum_{i=1}^{N} \ln \left[P(\mathcal{D}_i|z_i) P(z_i|\theta) \right] Q(\theta) d\theta \\
\Rightarrow \sum_i \ln Q(z_i) &\propto \int \sum_{i=1}^{N} \ln \left[P(\mathcal{D}_i|z_i) P(z_i|\theta) \right] Q(\theta) d\theta \\
\Rightarrow \ln Q(z_i) &\propto \int \ln \left[P(\mathcal{D}_i|z_i) P(z_i|\theta) \right] Q(\theta) d\theta && \text{(A.14)}
\end{aligned}$$

Taking exponential of both sides of eq. (A.14) we get :

$$Q(z_i) \propto \exp\left(\ln P(\mathcal{D}_i|z_i)\right) \exp\left(\left\langle \ln P(z_i|\theta) \right\rangle_{Q(\theta)} \right) \tag{A.15}$$

where $\langle . \rangle_Q$ denotes the expectation with respect to the distribution Q. When $P(z_i|\theta)$ belongs to the exponential family of distribution the term $\left\langle \ln P(z_i|\theta) \right\rangle_{Q(\theta)}$ results in a distribution belonging to the same family and as a result the posterior over the hidden variable $Q(z_i)$ is given by :

$$Q(z_i) \propto P(\mathcal{D}_i|z_i) P(z_i|\hat{\theta})$$

where $P(z_i|\hat{\theta}) = \exp\left(\left\langle \ln P(z_i|\theta) \right\rangle_{Q(\theta)} \right)$ is a distribution conjugate to $P(\mathcal{D}_i|z_i)$. This leads to a closed form expression for the distribution of the posterior $Q(z_i)$.

We can see that the end result of the VBE step is an approximate posterior for the hidden variable. In the next step of the procedure we use this approximate posterior over the hidden variable to estimate the posterior over the parameter θ.

A.2 VBM step

In this step we estimate the posterior over the parameter assuming that the estimate of the posterior over the hidden variable is known and is equal to the current estimate from the VBE step. We use the same strategy as before to obtain the estimate of the posterior. We take the functional derivative of $\mathcal{F}_{approx}$ with respect to $Q(\theta)$ and equate it to zero to obtain :

$$\begin{aligned}\frac{\partial \mathcal{F}_{approx}}{\partial Q(\theta)} &= \frac{\partial \int \left[\int \ln P(\mathbf{D},\mathbf{z}|\theta) + \ln \frac{P(\theta|\alpha}{Q(\theta)}\right] Q(\theta) d\theta}{\partial Q(\theta)} \\ &= \int \ln P(\mathbf{D},\mathbf{z}|\theta) Q(\mathbf{z}) d\mathbf{z} + \ln P(\theta|\alpha - \ln Q(\theta) + c \\ &= 0\end{aligned}$$

Solving for the equation we get :

$$\ln Q(\theta) \propto \ln P(\theta|\alpha) + \langle \ln P(\mathbf{D},\mathbf{z}|\theta)\rangle_{Q(\mathbf{z})} \tag{A.16}$$

Applying exponential on both sides and using eq. (A.1) and (A.2), eq. (A.16) becomes :

$$\begin{aligned}Q(\theta) &\propto \exp\left(\left\langle \sum_i \ln P(\mathcal{D}_i, z_i|\theta)\right\rangle_{Q(z_i)}\right) P(\theta|\alpha) \\ &\propto \exp\left(\sum_i \langle \ln P(\mathcal{D}_i, z_i|\theta)\rangle_{Q(z_i)}\right) P(\theta|\alpha) \\ &\propto \prod_i \exp\left(\langle \ln P(\mathcal{D}_i, z_i|\theta)\rangle_{Q(z_i)}\right)\end{aligned}$$

We can make the same arguments as for the VBE step and show that if the distributions involved belong to the conjugate exponential family, the posterior over the parameter has a closed form expression and belongs to the exponential family as well.

The two steps of VBE and VBM are repeated iteratively till convergence. Furthermore, it can be proven that this procedure is guaranteed to minimize the KL divergence between the actual joint posterior distribution $P(\mathbf{z},\theta|\mathbf{D})$ and the approximate posterior distribution $Q(\mathbf{z})Q(\theta)$.

Appendix B

Derivation of VBEM posteriors for RVC

We start with the approximate lower bound of the loglikelihood given by :

$$\begin{aligned}\mathcal{F}_{approx} = \sum_i \left[\langle \ln P(y_i|\beta_i,\sigma)\rangle_{Q_{\beta_i},Q_{\sigma^2}} + \left\langle \ln P(\beta_i|\hat{\beta},h_1\ldots h_{d+1})\right\rangle_{Q_{\beta_i},Q_{h_1}\ldots Q_{h_{d+1}},Q_{\hat{\beta}}}\right] \\ + \sum_j \left\langle \ln P(h_j^2|a_j,b_j)\right\rangle_{Q_{h_j}} + \left\langle \ln P(\hat{\beta}|\mu,\mathbf{S})\right\rangle_{Q_{\hat{\beta}}} + \left\langle \ln P(\sigma^2|c,d)\right\rangle_{Q_{\sigma^2}} \\ - \sum_i \left\langle \ln Q_{\beta_i}\right\rangle_{Q_{\beta_i}} - \sum_j \left\langle \ln Q_{h_j}\right\rangle_{Q_{h_j}} - \left\langle \ln Q_{\hat{\beta}}\right\rangle_{Q_{\hat{\beta}}} - \langle \ln Q_{\sigma^2}\rangle_{Q_{\sigma^2}}\end{aligned} \tag{B.1}$$

B.1 Derivation of Q_{β_i}

The form for the posterior over β_i can be derived by performing a functional differentiation of $\mathcal{F}_{approx}$ with respect to Q_{β_i} and equating it to zero. This would yield :

$$\frac{\partial \mathcal{F}_{approx}}{\partial Q_{\beta_i}} = \langle \ln P(y_i|\beta_i,\sigma)\rangle_{Q_{\sigma^2}} + \left\langle \ln P(\beta_i|\hat{\beta},\mathbf{h})\right\rangle_{Q_{\mathbf{h}},Q_{\hat{\beta}}} - \ln Z - \ln Q_{\beta_i} = 0 \tag{B.2}$$

$$\Rightarrow \ln Q_{\beta_i} = \langle \ln P(y_i|\beta_i,\sigma)\rangle_{Q_{\sigma^2}} + \left\langle \ln P(\beta_i|\hat{\beta},\mathbf{h})\right\rangle_{Q_{\mathbf{h}},Q_{\hat{\beta}}} - \ln Z \tag{B.3}$$

where Z is the normalization term for the posterior probability. Expanding eq. (B.3) by taking the expectations with respect to the posteriors of the other parameters we get :

$$\begin{aligned}\ln Q_{\beta_i} &\propto -\frac{1}{2}(y_i - \beta_i^T \mathbf{x}_i)^2 \left\langle 1/\sigma^2\right\rangle - \frac{1}{2}(\beta_i - \tilde{\mu})^T \langle \mathbf{C}_i\rangle^{-1} (\beta_i - \tilde{\mu}) \\ \Rightarrow Q_{\beta_i} &\sim \mathcal{N}(\nu_i, \mathbf{G}_i) \quad \text{with} \\ \mathbf{G}_i &= (\mathbf{x}_i\mathbf{x}_i^T \left\langle \frac{1}{\sigma^2}\right\rangle + \langle \mathbf{C}_i\rangle^{-1})^{-1} \\ \nu_i &= \mathbf{G}_i(y_i\mathbf{x}_i \left\langle \frac{1}{\sigma^2}\right\rangle + \langle \mathbf{C}_i\rangle^{-1}\tilde{\mu})\end{aligned}$$

where, $\langle \mathbf{C}_i \rangle = diag(\mathbf{x}_i^T \mathbf{x}_i / \left\langle h_j^2 \right\rangle_{Q(h_j^2)})$ and $\left\langle \frac{1}{\sigma^2} \right\rangle = \tilde{c}/\tilde{d}$.

B.2 Derivation of Q_{σ^2}

Performing a functional differentiation of $\mathcal{F}_{approx}$ with respect to Q_{σ^2} and equating it to zero would yield :

$$\frac{\partial \mathcal{F}_{approx}}{\partial Q_{\sigma^2}} = \sum_i \langle \ln P(y_i|\beta_i,\sigma) \rangle_{Q_{\beta_i}} + \ln P(\sigma^2|c,d) - \ln Z - \ln Q_{\sigma^2} = 0 \tag{B.4}$$

$$\Rightarrow \ln Q_{\sigma^2} = \sum_i \langle \ln P(y_i|\beta_i,\sigma) \rangle_{Q_{\beta_i}} + \ln P(\sigma^2|c,d) - \ln Z \tag{B.5}$$

Taking expectations and expanding eq. (B.5) yields :

$$\begin{aligned}
\ln Q_{\sigma^2} &\propto \sum_i \left[-\frac{1}{2}\ln\sigma^2 - \frac{1}{2}(y_i - \nu_i^T \mathbf{x}_i)^2/\sigma^2 - \frac{1}{2}\mathbf{x}_i^T \mathbf{G}_i \mathbf{x}_i/\sigma^2 \right] - (c+1)\ln\sigma^2 - d/\sigma^2 \\
\Rightarrow Q_{\sigma^2} &\sim IG(\tilde{c},\tilde{d}) \quad \text{with} \\
\tilde{c} &= c + N/2 \\
\tilde{d} &= d + \sum_i \left[(y_i - \nu_i^T \mathbf{x}_i)^2 + \mathbf{x}_i^T \mathbf{G}_i \mathbf{x}_i \right]/2
\end{aligned}$$

B.3 Derivation of $Q_{\hat{\beta}}$

Equating to the zero, the functional differential of $\mathcal{F}_{approx}$ with respect to $Q_{\hat{\beta}}$ yields :

$$\frac{\partial \mathcal{F}_{approx}}{\partial Q_{\hat{\beta}}} = \sum_i \left\langle \ln P(\beta_i|\hat{\beta},\mathbf{C}_i) \right\rangle_{Q_{\beta_i},Q_{\mathbf{h}}} + \ln P(\hat{\beta}|\mu,\mathbf{S}) - \ln Z - \ln Q_{\hat{\beta}} = 0 \tag{B.6}$$

$$\Rightarrow \ln Q_{\hat{\beta}} = \sum_i \left\langle \ln P(\beta_i|\hat{\beta},\mathbf{C}_i) \right\rangle_{Q_{\beta_i},Q_{\mathbf{h}}} + \ln P(\hat{\beta}|\mu,\mathbf{S}) - \ln Z \tag{B.7}$$

Taking expectations and expanding eq. (B.7) yields :

$$\begin{aligned}
\ln Q_{\hat{\beta}} &\propto \sum_i \left[-\frac{1}{2}(\beta_i - \hat{\beta})^T \langle \mathbf{C}_i \rangle^{-1} (\beta_i - \hat{\beta}) \right] - \frac{1}{2}(\hat{\beta} - \mu)^T \mathbf{S}^{-1} (\hat{\beta} - \mu) \\
\Rightarrow Q_{\hat{\beta}} &\sim \mathcal{N}(\tilde{\mu},\tilde{\mathbf{S}}) \quad \text{with} \\
\tilde{\mathbf{S}} &= (\sum_i \langle \mathbf{C}_i \rangle^{-1} + \mathbf{S}^{-1})^{-1} \\
\tilde{\mu} &= \tilde{\mathbf{S}} (\sum_i \langle \mathbf{C}_i \rangle^{-1} \nu_i + \mathbf{S}^{-1} \mu)
\end{aligned}$$

B.4 Derivation of Q_{h_j}

The diagonal structure of $\mathbf{C}_i$ ensures that the posterior over $Q_{\mathbf{h}}$ factorizes into individual components Q_{h_j}. Performing a functional differentiation of $\mathcal{F}_{approx}$ with respect to

Q_{h_j} and equating it to zero would yield :

$$\frac{\partial \mathcal{F}_{approx}}{\partial Q_{h_j}} = \sum_i \left\langle \ln P(\beta_i|\hat{\beta}, \mathbf{C}_i) \right\rangle_{Q_{\beta_i}, Q_{\hat{\beta}}} + \ln P(h_j^2|a_j, b_j) - \ln Z - \ln Q_{h_j} = 0 \quad \text{(B.8)}$$

$$\Rightarrow \ln Q_{h_j} = \sum_i \left\langle \ln P(\beta_i|\hat{\beta}, \mathbf{C}_i) \right\rangle_{Q_{\beta_i}, Q_{\hat{\beta}}} + \ln P(h_j^2|a_j, b_j) - \ln Z \quad \text{(B.9)}$$

Taking expectations and expanding eq. (B.9) yields :

$$\begin{aligned}
\ln Q_{h_j} &\propto \sum_i \left[\frac{1}{2} \ln h_j^2 - \frac{1}{2} \frac{(\nu_{i,j} - \tilde{\mu}_{i,j})^2 + \mathbf{G}_{i,jj} + \tilde{\mathbf{S}}_{jj}}{\mathbf{x}_i^T \mathbf{x}_i} h_j^2 \right] + (a_j - 1) \ln h_j^2 - b_j h_j^2 \\
\Rightarrow Q_{h_j} &\sim \mathcal{G}(\tilde{a}_j, \tilde{b}_j) \quad \text{with} \\
\tilde{a}_j &= a_j + N/2 \\
\tilde{b}_j &= b_j + \sum_i \left[(\nu_{i,j} - \tilde{\mu}_{i,j})^2 + \mathbf{G}_{i,jj} + \tilde{\mathbf{S}}_{jj} \right] / (2 \mathbf{x}_i^T \mathbf{x}_i)
\end{aligned}$$

Appendix C

Approximation of predictive distribution

We approximate predictive distribution given in eq. (4.16) by $P(y_q|\tilde{\mu}^T\mathbf{x}_q, v)$ where,

$$P(y_q|\tilde{\mu}^T\mathbf{x}_q, v) \sim \mathcal{N}(\tilde{\mu}^T\mathbf{x}_q, v^2)$$

The KL divergence between the two distributions is given by,

$$\int \left[\int P(y_q|\tilde{\mu}, \sigma, \mathbf{h}) Q(\mathbf{h}|\mathbf{y}) Q(\sigma|\mathbf{y}) d\mathbf{h} d\sigma \right] \ln \frac{\int P(y_q|\tilde{\mu}, \sigma, \mathbf{h}) Q(\mathbf{h}|\mathbf{y}) Q(\sigma|\mathbf{y}) d\mathbf{h} d\sigma}{P(y_q|\tilde{\mu}^T\mathbf{x}_q, v)} dy_q \tag{C.1}$$

where,

$$P(y_q|\tilde{\mu}, \sigma, \mathbf{h}) \sim \mathcal{N}(\tilde{\mu}^T\mathbf{x}_q, \mathbf{x}_q^T(C_\mathbf{h} + \tilde{\mathbf{S}})\mathbf{x}_q + \sigma^2) \tag{C.2}$$

We need to optimise eq. (C.1) with respect to v. Writing down the term dependent on v and exchanging the order of integration we get,

$$\begin{aligned} \mathcal{L} &= -\int \left[\int P(y_q|\tilde{\mu}, \sigma, \mathbf{h}) \ln P(y_q|\tilde{\mu}^T\mathbf{x}_q, v) dy_q \right] Q(\mathbf{h}|\mathbf{y}) Q(\sigma|\mathbf{y}) d\mathbf{h} d\sigma \\ &= -\left\langle \int P(y_q|\tilde{\mu}, \sigma, \mathbf{h}) \ln P(y_q|\tilde{\mu}^T\mathbf{x}_q, v) dy_q \right\rangle_{Q(\mathbf{h}|\mathbf{y}), Q(\sigma|\mathbf{y})} \\ &= \frac{1}{2}\ln v^2 + \frac{1}{2}\frac{\mathbf{x}_q^T(\langle C_\mathbf{h}\rangle_{Q(\mathbf{h}|\mathbf{y})} + \tilde{\mathbf{S}})\mathbf{x}_q + \langle\sigma^2\rangle_{Q(\sigma|\mathbf{y})}}{v^2} \end{aligned} \tag{C.3}$$

where,

$$\begin{aligned} \langle C_\mathbf{h}\rangle_{Q(\mathbf{h}|\mathbf{y})} = \quad & diag(x_q^T x_q \left\langle \tfrac{1}{h_j^2} \right\rangle_{Gamma(h_j^2)}) = \quad C_{\mathbf{h}_{mode}} \\ \langle\sigma^2\rangle_{Q(\sigma|\mathbf{y})} = \quad & \langle\sigma^2\rangle_{InvGamma(\sigma^2)} = \quad \sigma^2_{mean} \end{aligned}$$

Differentiating $\mathcal{L}$ w.r.t to v and equating to zero we get $v^2 = \mathbf{x}_q^T(C_{\mathbf{h}_{mode}} + \tilde{\mathbf{S}})\mathbf{x}_q + \sigma^2_{mean}$.

Bibliography

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*, 716–723.

Allen, D. M. (1974). The relationship between variable selection and data augmentation and a method of prediction. *Technometrics*, *16*, 125–127.

Atkeson, C., Moore, A., & Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, *11*, 76–113.

Beal, M. J. (2003). *Variational algorithms for approximate Bayesian inference*. Doctoral dissertation, Gatsby Computational Neuroscience Unit, University College London.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Claredon Press.

Bishop, C. M., & Svensèn, M. (2003). Bayesian hierarchical mixture of experts. *Uncertainty in Artificial Intelligence*.

Brown, G., Wyatt, J., Harris, R., & Yao, X. (2005a). Diversity creation methods: A survey and categorisation. *Journal of Information Fusion*, *6*, 5–20.

Brown, G., Wyatt, J. L., & Tino, P. (2005b). Managing diversity in regression ensembles. *Journal of Machine Learning Research*, *6*, 1621–1650.

Bruntz, S. M., Cleveland, W. S., Kleiner, B., & Warner, J. L. (1974). The dependence of ambient ozone on solar radiation, temperature and mixing height. *Symp. Atmospheric Diffusion and Air pollution*. MIT Press.

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Choi, E., & Hall, P. (1998). On bias reduction in local linear smoothing. *Biometrika*, *85*, 333–345.

Cleveland, W. S., & Loader, C. (1995). *Smoothing by local regression: Principles and methods* (Technical Report). AT&T Bell Laboratories.

Csatò, L., & Opper, M. (2002). Sparse on-line Gaussian processes. *Neural Computation*, *14*, 641–668.

De Forest, E. L. (1873). On some methods of interpolation applicable to the graduation of irregular series. *Annual Report of the Board of Regents of the Smithsonian Institution for 1871*, 275–339.

De Forest, E. L. (1874). Additions to a memoir on methods of interpolation applicable to the graduation of irregular series. *Annual Report of the Board of Regents of the Smithsonian Institution for 1873*, 319–353.

Demirekler, M., & Altinçay, H. (2002). Plurality voting-based multiple classifier systems: statistically independent with respect to dependent classifier sets. *Pattern Recognition*, *35*, 2365–2379.

Dietterich, T. G. (2002). Ensemble learning. In M. Arbib (Ed.), *The handbook of brain theory and neural networks*, 405–408. The MIT Press.

Edakunni, N. U., Schaal, S., & Vijayakumar, S. (2007). Kernel carpentry for online regression using randomly varying coefficient model. *International Joint Conference on Artificial Intelligence* (pp. 762–767).

Edakunni, N. U., & Vijayakumar, S. (2009). Efficient online classification using an ensemble of bayesian linear logistic regressors. *Multiple Classifier Systems* (pp. 102–111).

Fan, J., & Gijbels, I. (1995). Adaptive order polynomial fitting: Bandwidth robustification and bias reduction. *Journal of Computational and Graphical Statistics*, *4*, 213–227.

Fan, J., & Gijbels, I. (1996). *Local polynomial modelling and its applications*. Chapman and Hall.

Figueiredo, M. A. (2003). Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*, 1150–1159.

Freund, Y., & Schapire, R. (1996). Game theory, online prediction and boosting. *Ninth Annual Conference on Computational Learning Theory* (pp. 325–332).

Gasser, T., Kneip, A., & Köhler, W. (1991). A flexible and fast method for automatic smoothing. *Journal of American Statistical Association*, *86*, 643–652.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2003). *Bayesian data analysis*. Chapman and Hall/CRC. Second edition.

Gelman, A., & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press.

Gelman, A., & Pardoe, I. (2006). Bayesian measures of explained variance and pooling in multilevel (hierarchical) models. *Technometrics*, *48*, 241–251.

Gram, J. P. (1883). Über entwickelung reeller functionen in reihen mittelst der methode der kleinsten quadrate. *Journal für Mathematics*, *94*, 41–73.

Härdle, W. (1994). *Applied nonparametric regression*. Cambridge University Press.

Hashem, S., & Schmeiser, B. (1995). Improving model accuracy using optimal linear combinations of trained neural networks. *IEEE Transactions on Neural Networks*, *6*, 792–794.

Hastie, T., & Loader, C. (1993). Local regression: Automatic kernel carpentry. *Statistical Science*, *8*, 120–129.

Hastie, T. J., & Tibshirani., R. J. (1990). *Generalized additive models.* Chapman and Hall.

Hinton, G. E. (1999). Product of experts. *Ninth International Conference on Artificial Neural Networks*.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554.

Hoffmann, H., Schaal, S., & Vijayakumar, S. (2009). Local dimensionality reduction for nonparametric regression. *Neural Processing Letters*, *29*, 109–131.

Jones, M., Marron, J., & Sheather, S. (1996). A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association*, *91*.

Jones, M. C., Davies, S. J., & Park, B. U. (1994). Versions of kernel-type regression estimators. *Journal of the American Statistical Association*, *89*, 825–832.

Kittler, J., Hatef, M., Duin, R. P. W., & Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 226–239.

Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. *Advances in Neural Information Processing Systems* (pp. 231–238).

Kschischang, F. R., Frey, B. J., & Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*, 498–519.

Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with ensemble accuracy. *Machine Learning*, *51*, 181–207.

Kuncheva, L. I., Whitaker, C. J., Shipp, C. A., & Duin, R. P. W. (2000). Is independence good for combining classifiers ? *International Conference on Pattern Recognition*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *International Conference on Machine Learning*.

Loader, C. (1999a). *Local regression and likelihood*. Springer-Verlag.

Loader, C. R. (1999b). Bandwidth selection : Classical or plug-in ? *The Annals of Statistics*, *27*, 415–438.

Longford, N. T. (1993). *Random coefficient models*. Clarendon Press.

MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation*, *4*, 415–447.

Mallows, C. L. (1973). Some comments on c_p. *Technometrics*, *15*, 661–675.

Murray, I., & Ghahramani, Z. (2004). Bayesian learning in undirected graphical models: Approximate mcmc algorithms. *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability and its Applications*, *9*, 141–142.

Opper, M. (1998). A Bayesian approach to on-line learning. In *On-line learning in neural networks*, 363–378. New York, NY, USA: Cambridge University Press.

Price, P. N., Nero, A. V., & Gelman, A. (1996). Bayesian prediction of mean indoor radon concentrations for minnesota counties. *Health Physics*, *71*, 922–936.

Qi, Y. A., & Jaakkola, T. S. (2006). Parameter expanded variational bayesian methods. *Advances in Neural Information Processing Systems*. MIT Press.

Rasmussen, C. E., & Gharamani, Z. (2000). Infinite mixtures of Gaussian process experts. *Advances in Neural Information Processing Systems 14*. MIT Press.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Ripley, B. D. (1996). *Pattern recognition and neural networks*. Cambridge University Press.

Ruppert, D., Sheather, S. J., & Wand, M. P. (1995). An effective bandwidth selector for local least squares regression. *Journal of the American Statistical Association*, *90*.

Sato, M. (2001). Online model selection based on the variational Bayes. *Neural Computation*, *13*, 1649–1681.

Schaal, S., & Atkeson, C. (1998). Constructive incremental learning from only local information. *Neural Computation*, *10*, 2047–2084.

Schapire, R. E. (1999). A brief introduction to boosting. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*.

Silverman, B. W. (1984). Spline smoothing : The equivalent variable kernel method. *The Annals of Statistics*, *12*, 898–916.

Smola, A., & Bartlett, P. (2001). Sparse greedy Gaussian process regression. *Advances in Neural Information Processing Systems 13* (pp. 619–625). MIT Press.

Snelson, E., & Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems 18* (pp. 1257–1264). MIT press.

Sollich, P., & Williams, C. K. I. (2005). Using the equivalent kernel to understand gaussian process regression. *Advances in Neural Information Processing Systems*.

Spencer, J. (1904). On the graduation of the rates of sickness and mortality. *Journal of the Institute of Actuaries*, *38*, 334–347.

Sutton, C., & McCallum, A. (2005). Piecewise training of undirected models. *21st Conference on Uncertainty in Artificial Intelligence*.

Szegö, G. (1992). *Orthogonal polynomials*. American Mathematical Society.

Tierney, L., & Kadane, J. (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, *81*.

Tipping, M. E., & Lawrence, N. D. (2005). Variational inference for Student-t models: Robust Bayesian interpolation and generalised component analysis. *Neurocomputing*, *69*, 123–141.

Tresp, V. (2000). A Bayesian committee machine. *Neural Computation*, *12*, 2719–2741.

Vapnik, V. N. (1998). *Statistical learning theory*. Wiley, New York.

Vijayakumar, S., D'Souza, A., & Schaal, S. (2005). Incremental online learning in high dimensions. *Neural Computation*, *17*.

Vijayakumar, S., D'Souza, A., Shibata, T., Conradt, J., & Schaal, S. (2002). Statistical learning for humanoid robots. *Autonomous Robot*, *12*, 55–69.

Waterhouse, S., Mackay, D., & Robinson, T. (1996). Bayesian methods for mixture of experts. *Advances in Neural Information Processing Systems* (pp. 351–357). MIT Press.

Watson, G. S. (1964). Smooth regression analysis. *Sankhya Ser. A*, *26*, 359–372.

Williams, C. K. I. (1998). Prediction with Gaussian processes: from linear regression to linear prediction and beyond. In M. I. Jordan (Ed.), *Learning in graphical models*, 599–621. Kluwer.

Williams, C. K. I., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems 13* (pp. 682–688). MIT Press.

Woodrofe, M. (1970). On choosing a delta-sequence. *Annals of Mathematical Statistics*, *41*, 1665–1671.

Woolhouse, W. S. B. (1870). Explanation of a new method of adjusting mortality tables, with some observations upon Mr. Makeham's modification of Gompertz's theory. *Journal of the Institute of Actuaries*, *15*, 389–410.

Xu, L., Hinton, G., & Jordan, M. I. (1995). An alternative model for mixtures of experts. *Advances in Neural Information Processing Systems*, *7*, 633–640.