TOKYO INSTITUTE OF TECHNOLOGY

TOKYO INSTITUTE OF TECHNOLOGY

Computational Theory of Incremental and Active Learning for Optimal Generalization

Sethu Vijayakumar

Submitted in partial fulfillment of requirements for the degree of

Doctor of Philosophy (CS&E.)

Department of Computer Science,
Graduate School of Information Science and Engineering,
Tokyo Institute of Technology, Tokyo.

January 1998

Supervisor: **Prof. Dr. Hidemitsu Ogawa**

Support: Financial support to the author was provided by the Japanese Ministry of Education, Science and Culture

©Sethu Vijayakumar 1998

©Sethu Vijayakumar 1998

To my parents

©Sethu Vijayakumar 1998

©Sethu Vijayakumar 1998

Contents

I Introduction **1**

Notation **3**

1 Learning about Learning **5**

1.1 Introduction 5

1.2 Learning and generalization 6

1.3 Statistical, Computational and Neuro-biological approaches 7

1.4 Computational learning theory 9

1.4.1 What encompasses this field? 9

1.4.2 Neural Networks as models for learning 10

1.4.3 The present and the shortcomings 11

1.5 The What and Why of this research work 13

1.6 Organization of this dissertation 14

II Exact Incremental and Active Learning **17**

2 Functional Analytic Approach to Neural Network Learning **19**

2.1 Introduction 19

2.2 Learning as an inverse problem 20

2.3 Generalization ability and optimization criterion 23

2.3.1 Generalization ability 23
2.3.2 Memorization criterion 23
2.3.3 Wiener criterion 26
2.3.4 Projection criterion 27
2.4 Realization of optimally generalizing NN (OGNN) 29

3 Model Selection **35**
3.1 Introduction 35
3.2 Standard model selection strategies 36
3.3 The bias-variance dilemma 37
3.4 Using family of stratified Hilbert spaces 39
3.4.1 Hilbert space and reproducing kernel 39
3.4.2 Some useful Hilbert spaces and their formulation 40

4 Exact Incremental Learning for Optimal Generalization **45**
4.1 Introduction 45
4.2 The concept of incremental learning 46
4.3 Moore-Penrose generalized inverse 49
4.4 Geometric interpretation of Wiener learning 51
4.5 The incremental Wiener criterion 53
4.6 Incremental computation of the learning operator 54
4.7 Incremental computation of the learned function 59
4.8 General incremental learning results 61
4.9 Pseudo code of incremental learning algorithm 62
4.10 Simulation results and experiments 64
4.10.1 The generalization capability 65
4.10.2 The effect of training data location 67
4.10.3 The model selection problem 69
Appendix A4 72

5 Learning sensorimotor maps using apriori knowledge **77**
5.1 Introduction 77
5.2 Equations of dynamics of motion 78

©Sethu Vijayakumar 1998

5.3 Hilbert search space for 2-DOF robot arm 81
5.4 Experiments in learning of the sensorimotor map 82
5.4.1 Empirical evaluation results and learning curves 83
5.4.2 The variance of solution based on training data location . . . 85
Appendix A5 88

6 Active Data Selection for Optimal Generalization 97
6.1 The active learning framework 97
6.2 Various information measure and approaches 98
6.3 The Wiener criterion based information measure 99
6.3.1 Criterion for batch selection of complete data set 100
6.3.2 Criterion for constrained optimal selection of next data . . . 101
6.4 Data selection strategy 101
6.5 Comments 105
Appendix A6 107

III Incremental Learning in High Dimensional Spaces 109

7 Local Dimensionality Reduction 111
7.1 The curse of dimensionality 111
7.2 Evidence towards locally low dimensional structure in motion data . 112
7.2.1 The dimensionality analysis tool 112
7.2.2 The dimensionality analysis results 114
7.3 Comparison of local dimensionality reduction techniques for regression 116
7.3.1 Model used for comparison 117
7.3.2 Locally weighted factor analysis(LWFA) 118
7.3.3 Locally weighted principal component regression(LWPCR) . 119
7.3.4 Locally weighted principal component analysis in joint data space (LWPCA) 120
7.3.5 Locally weighted partial least squares(LWPLS, LWPLS1) . . 120
7.3.6 Monte Carlo evaluations for comparison 122

©Sethu Vijayakumar 1998

7.4 Comments 125

8 Locally Weighted Learning with LASS 127

8.1 Introduction 127

8.2 The statistical model of LASS 129

8.3 Locally weighted dimensionality reduction in LASS (LWPCA) 132

8.3.1 The basic theory 132

8.3.2 Locally weighted and incremental implementation 133

8.4 Locally weighted regression (LWR) in LASS 135

8.5 Distance metric adaptation in LASS 137

8.6 The working of the LASS algorithm 141

8.7 LASS pseudocode implementation 143

9 LASS High Dimensional Learning Performance 145

9.1 Introduction 145

9.2 Learning in 10-dimensional artificial example 146

9.3 Learning a 21-dimensional sensorimotor map 148

9.3.1 Sensorimotor map of a 7-DOF anthropomorphic robot arm . 148

9.3.2 Trajectory tracking performance in a drawing 8 task 149

9.4 Learning with real human arm data 149

9.5 Comments 151

IV Conclusion 153

10 Conclusions & Future Objectives 155

10.1 Conclusions 155

10.2 Future Objectives 157

Acknowledgments 159

Bibliography 161

©Sethu Vijayakumar 1998

List of Figures

1.1 Learning from statistical, computational and neuro-biological perspectives 8

1.2 General neural network architecture 10

2.1 NN as a real valued function 20

2.2 NN learning as an inverse problem 22

2.3 Memorization learning : Optimization in the sampled space 24

2.4 Projection learning 28

2.5 Conventionally used NN architecture 29

2.6 nth hidden unit of the neural network 30

2.7 NN architecture used in this dissertation 31

2.8 Functional analytic approach to NN learning. 32

3.1 Reproducing kernels for 2-dimensional band-limited function spaces with cutoff frequency parameters (a) $\sigma_1 = 8, \sigma_2 = 8$ and (b) $\sigma_1 = 10, \sigma_2 = 3$. Note the radial asymmetry in (b). 41

3.2 Function approximation using the projection learning criterion and using original function spaces with frequency cut-off (b) $\sigma = 3$, (c) $\sigma = 10$ and (d) $\sigma = 40$ 42

3.3 Legendre Polynomials : An example of orthogonal polynomials of increasing degree 43

3.4 Reproducing kernels $K(x,x')$ for function spaces spanned by polynomial functions with (a) highest degree polynomial = 2 and (b) highest degree polynomial = 5 43

4.1 Neural network learning model. The two stages involved: (i) The selection of training data and hence, the sampling operator A, (ii) Evaluating the optimum learning operator X. 46

4.2 Incremental learning with additive training data 47

4.3 Geometrical interpretation of a Wiener learning. For an operator X satisfying the Wiener criterion, XA provides an oblique projection f_1 of $f \in \mathcal{R}(R)$ onto $\mathcal{R}(RA^*)$ along $\mathcal{R}(R) \cap \mathcal{N}(A)$. 53

4.4 Relationship between the Wiener and Projection learning operators. Wiener learning is an oblique projection onto $\mathcal{R}(RA^*)$ whereas Projection learning is an orthogonal projection onto $\mathcal{R}(A^*)$, both along $\mathcal{N}(A)$. 54

4.5 Operator Γ mapping $y^{(m)}$ from R^m into R^{m+1} space 55

4.6 Structure of the matrix U_{m+1} 56

4.7 Relationship between f_m and f_{m+1}. 60

4.8 Incremental function approximation using the projection learning with (a)one (b)two (c)four and (d)six training data. The original function is shown by a dotted line and the learned function by a solid line. The training data is marked by tiny squares. 66

4.9 Effect of training data location on incremental learning results. The original function f(dashed), the function learned with four data points f_4(dot-dashed) and the function learned with five data points f_5(solid) are plotted. (a) shows learning with fifth data point selected in a redundant fashion; here f_4 (dot-dashed line) and f_5 (solid line) coincide (b) shows results with fifth data point chosen optimally. 68

4.10 Function approximation result for original function $f = -2sin(x) + 3cos(x) + sin(2x) + 4cos(2x) - 0.5sin(3x)$ with one (f_1), two (f_2) and three (f_3) sample points. (a) uses a 6-dimensional Hilbert space H_1 while (b) uses an 8-dimensional Hilbert space H_2 70

5.1 An idealized model of a 2 DOF robot arm 78

©Sethu Vijayakumar 1998

5.2 The Mean Squared Error (MSE) on the training data(dashed line) and on a set of 100 test data(solid line) for the sensorimotor map learned with different number of training data; the bars represent the maximum and minimum values and the dot represents the mean of the MSE. 84

5.3 The actual generalization error (the norm of the difference between the true function and the learned function) for the sensorimotor map learned with different number of training data; the bars represent the maximum and minimum values and the dot represents the mean of the true error. 85

5.4 Variance of the approximation results with location of the training data 86

5.5 Idealized representation of 2DOF robot arm 90

7.1 (a) A synthetic 3 dimensional data set consisting of a quadratic surface. (b)The variation in local dimensionality with decreasing locality for this data. 113

7.2 Dimensionality of 2 DOF robot movement data performing random motion in 6 dimensional input space. 114

7.3 Dimensionality of 2 DOF robot movement data performing doing a specific task (here, drawing a figure "eight") in 6 dimensional input space. Note the further reduction in local dimensionality compared to random movement tasks. 115

7.4 Dimensionality of 7 DOF SARCOS anthropomorphic robot arm data. A drastic reduction in dimensionality from the 21 dimensional input space to about 10-12 dimensions locally is observed. 115

7.5 Average summary results of Monte Carlo experiments. Each chart is primarily divided into the three major noise conditions, cf. headers in chart (a). In each noise condition, there are four further subdivision: coefficients of linear or quadratic model are equal with (i) low added noise; (ii) with high added noise; coefficients of linear or quadratic model are different with (iii) low noise added; (iv) with high added noise. Refer to text and descriptions of Monte Carlo studies for further explanations. 123

8.1 Organization of local elements in LASS 130

8.2 Illustration of the information processing stages in one element of LASS. 131

8.3 An example of approximation of a complex function using receptive field based locally linear models 136

©Sethu Vijayakumar 1998

9.1 (a) Surface plot of function 'cross' to be approximated; (b) LASS approximation results for the 10-dimensional input data set created by projecting the 'cross' data onto 10 dimensions (8 of them redundant) and rotating (c) Receptive field in input space (projected back onto 2-D) after a few epochs of learning, given by contour lines of 0.1 isoactivation along with centers marked by a 'o' (training data displayed by small dots) (d) receptive fields after the completion of training 146

9.2 $nMSE$(solid lines) and dimensionality of regression(dashed lines) as function of training iterations for 10-D LASS approximation with fixed and adjusted distance metric and a baseline comparison with 2-D RFWR. One training iteration corresponds to one incremental presentation of a training sample. 147

9.3 (a) Sketch of SARCOS Dexterous Arm (b) Trajectory tracking comparison of the SARCOS arm using a PID controller against using a non-parametric inverse dynamics model learned by LASS. 149

9.4 Average dimensionality and typical learning curves for LASS in real movement data of humans and robots. 150

©Sethu Vijayakumar 1998

List of Tables

4.1 Incremental computation of learning operators for various optimization criterion 63

4.2 Incrementally learned functions under different optimization criteria 63

6.1 Types of active learning 100

©Sethu Vijayakumar 1998

Part I

Introduction

Notation

Notation	Affectation
NN	Neural Network
x	Input Vector
x_m	Sample point
$\{x_m\}_{m=1}^{M}$	Training set
y_m	Sample value
y	Sample vector
$y^{(m)}$	Sample vector using m sample values
$\{x_m, y_m\}_{m=1}^{M}$	Set of training data
$\{e_m\}_{m=1}^{M}$	Natural basis
$\{e_n^{(m)}\}_{n=1}^{m}$	Natural basis of m dimensional vector space
H	Hilbert space of functions to be approximated
R^m	Sampled space of dimension m
A	Sampling operator
A_m	Sampling operator for m sample points
X	Learning operator
X_m	Learning operator corresponding to A_m
R	Correlation operator of functions to be approximated
$(.,.)$	Inner product
$\bar{.}$	complex conjugate
$. \otimes \bar{.}$	Schatten product
u	Vector u
u^T	Transpose of vector u
A^*	Adjoint operator of A
$A^\dagger$	Moore-Penrose generalized inverse of A

$\mathcal{R}(A)$	Range of A
$\mathcal{N}(A)$	Null space of A
$rank(A)$	Rank of matrix A
$dim(S)$	Dimension of space S
$tr(A)$	Trace of A
$S^{\perp}$	Orthogonal complement of subspace S
$\|A\|$	Norm of operator A
I	Identity matrix
$f(x)$	Original function to be approximated
$f_0(x)$	Learned function
$f_m(x)$	Learned function due to m sample points
J_M	Memorization learning criterion
J_W	Wiener learning criterion
J_P	Projection learning criterion
DOF	Degree of Freedom

Subscript/Superscript usage norms in Part II

x	The scalar x (lower case italics)
x_m	The m-th instance of the scalar x.
$\mathbf{x}$	The vector $\mathbf{x}$ (lower case bold face)
$\mathbf{x} \equiv (x_i)$	The elementwise breakup of the vector $\mathbf{x}$
$\mathbf{x}_m$	The m-th instance of the vector $\mathbf{x}$
$\mathbf{x}_m \equiv (x_{m,i})$	The elementwise breakup of the vector $\mathbf{x}_m$
$\mathbf{M}$	Matrix (upper case bold face)
$\mathbf{M} \equiv (M_{ij})$	The elementwise breakup of matrix $\mathbf{M}$
$x^{(n)}$	The value of the scalar x after n iterations
$\mathbf{x}^{(n)}$	The value of the vector $\mathbf{x}$ after n iterations
$\mathbf{M}^{(n)}$	The value of the matrix $\mathbf{M}$ after n iterations

©Sethu Vijayakumar 1998

1

Learning about Learning

1.1 Introduction

One of the outstanding characteristics of biological systems is their ability to learn, in particular, to learn incrementally in real time from a multitude of sensory inputs. Despite progress in artificial neural network learning, statistical learning and machine learning, we are still far away from equipping an artificial system of even moderate complexity with a 'black box' learning system which can perform as autonomously and robustly as their biological counterpart.

Looking at reasons for this inability, we see that on one hand, mathematicians and theoreticians have developed rigorous theories on learnability, PAC learning, complexity of learning tasks and the likes while mostly ignoring real world constraints and applicability. On the other hand, there has been a boom of application oriented research especially in the field of artificial neural network learning, which have focused on developing algorithms which work for a certain subset of problem class (or even smaller units) without really gouging the limits and constraints of the method being used. However, in this dissertation, I take the view that a clear definition of what is being aimed for is essential to start with. The functional analytic approach provides a framework which is used to directly formalize what kind of errors (and in what sense) are being minimized and what are the implicit assumptions we make when choosing a particular solution. A strong theoretical framework forms the backbone of the analysis which is extended in later parts to less exact (approximate) methods of learning in high dimensional cases which have very limited apriori knowledge. This chapter will give a working definition of the concept of *learning* as used in this work, provide a broad overview of the different approaches to the

study of learning and will chalk out the contributions of this thesis along with a description of the organization of the dissertation.

1.2 Learning and generalization

In the previous section, we have referred to the problem of "learning" a number of times. *Learning* per se is defined and interpreted in many different ways in different contexts [Val84, PG90, NT89]. Here, we will give a working definition of learning which we will use during the course of this dissertation. By "learning", we refer to the process of obtaining an approximation to the underlying unknown process (or function) based on a set of observables (or training data) which is generated by sampling the original process. As can be seen through this definition, our approach is very little biologically oriented and hence, motivational levels, development and other factors, which make analyzing biological models of learning difficult, are not an issue here.

Of course, correctly reproducing the training data can be usually done using a table look-up kind of method. Many memory based learning schemes working on those principles have been implemented [SP63]. However, the main aim of learning is to represent the concept behind the data and use more compact representation that models the generating process well. Moreover, this will ensure "good generalization capability", i.e. ability to respond correctly to novel inputs which are not part of the training data. Putting this in more concrete terms, if we are given a set of training data $\{x_m, y_m\}_{m=1}^{M}$, and we try and model this information by searching for an optimal approximation f_0 within a search space, then the errors over the training data is given by the *empirical risk*

$$S(f_0) = \frac{1}{M} \sum_{m=1}^{M} e(y_m, f_0(x_m)) \tag{1.1}$$

where $e(.,.)$ is some measure or error criterion under which we are optimizing our learning process. However, we are interested not only in reducing errors over the training set but also improving generalization. So, error over novel data sets have to be considered. This error, referred to as the *expected risk* or *generalization error*, is a measure over all possible input-output pairs of the system under consideration, and is given by

$$G(f_0) = e(f, f_0) = \int \int e(y, f_0(x)) p(y|x) p(x) dx dy. \tag{1.2}$$

Here, $p(x)$ refers to the probability distribution of x and $p(y|x)$ represents the conditional probability of y. This error is in general not tractable since we do not know the

©Sethu Vijayakumar 1998

original function f and it is impossible to sample for all the points in input domain. However, there are techniques (as will be shown in later chapters) to evaluate the optimal approximation functions within a certain domain without explicitly computing the generalization error. An analogue to this kind of learning can be found in robotics or biologically inspired learning systems, for example, in the area of motor control. The training data corresponds to the torque applied to various joints of (for example) an arm and the resultant motion parameters like the angular acceleration, velocity and position. Based on this, learning, which here corresponds to improving the sensorimotor map (also a multi dimensional highly non-linear function) occurs progressively with more and more experience (or data) improving task performance.

1.3 Statistical, Computational and Neuro-biological approaches

The goal of building systems that can adapt to their environments and learn from their experience has attracted researchers from many fields, including computer science, engineering, mathematics, physics, neuroscience, and cognitive science. Out of this research has come a wide variety of learning techniques, including methods for learning decision trees, decision rules, neural networks, statistical classifiers, and probabilistic graphical models. The researchers in these various areas have also produced several different theoretical frameworks for understanding these methods, such as computational learning theory, Bayesian learning theory [Bis95], classical statistical theory, minimum description length theory, and statistical mechanics approaches [OKC95]. These theories provide insight into experimental results and help to guide the development of improved learning algorithms.

On the other hand, there is the neuro-biological approaches which again is broadly classified under *wetware* experimental research and the modeling of neuronal activities by mathematical models to understand their mechanism. This research field mainly aims at contributing towards understanding the mechanism of information transfer and processing in the brain and putting forth biological constraints and conditions under which computational models can be evaluated as to whether they really model processes in the brain.

Hence, two mutually reinforcing objectives of neural modeling were defined and remain today: first, to understand the physiological and psychological functioning of the human neural system; and second, to produce computational systems that perform brainlike functions. Fig.1.1 highlights the major aims of statistical, computational and neuro-biological approaches and emphasizes the fact that all these

©Sethu Vijayakumar 1998

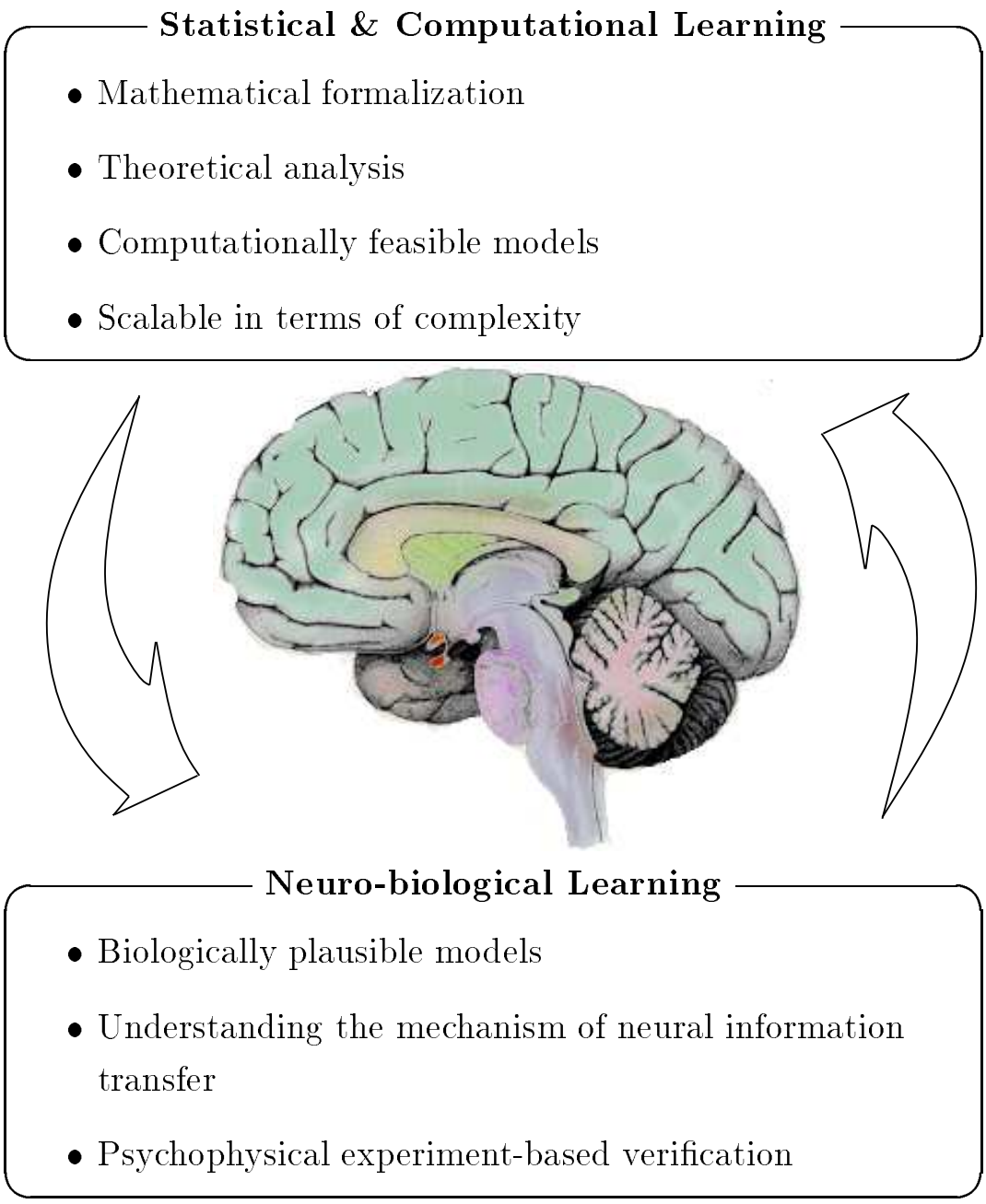


Figure 1.1: Learning from statistical, computational and neuro-biological perspectives

©Sethu Vijayakumar 1998

fields have to work in a cooperative sense in order to elucidate the working of the brain and improve our understanding of the concept of learning. The next section describes the views and standpoint taken in this dissertation.

1.4 Computational learning theory

In this dissertation, from among the various approaches that has been mentioned, we try to use the computational learning approach to comprehend the concept of learning and generalization. The point of view taken is that learning is an information processing problem : a learning system modifies commands (or outputs) using information from previous experiences so as to improve performance measured using some scale. Another analogous explanation is that learning deals with estimating the *system* that transforms inputs to outputs given a set of examples of the input-output pair.

1.4.1 What encompasses this field?

A proposed methodology for computational learning includes the following steps [Mar77]:

1. Identify a particular information processing problem or task.

2. Formulate a computational theory: Expose constraints that enable (or restrict) performance, describe what is being computed and why, select or devise appropriate representations, define procedures to solve problem.

3. Devise and implement actual algorithms to solve the problem.

4. Verify and evaluate the performance of algorithms via experiments.

It is Item 2 in the list that we concentrate on here. To model the learning process, artificial neural networks have been extensively used, mainly because of their historical connections with the model of the brain. The design of neural networks is motivated by analogy with the brain, which is a living proof that fault tolerant parallel processing is not only physically possible but is also fast and powerful.

©Sethu Vijayakumar 1998

1.4.2 Neural Networks as models for learning

Neurocomputing is the technological discipline concerned with parallel, distributed, adaptive information processing systems that develop information processing capabilities in response to exposure to an information environment. The primary information processing structures of interest in Neurocomputing are *neural networks.* The formal definition of a neural network follows.

Definition 1.1 (Neural Networks) *A* neural network *is a parallel, distributed information processing structure consisting of* processing elements*, which can possess a local memory and carry out localized information processing operations, interconnected via unidirectional signal channel called* connections*. Each processing element has a single output connection that branches or "fans out"into as many collateral connections as desired; each carries the same signal referred to as* the processing element output signal*. The processing element output signal can be of any mathematical type desired. The information processing that goes on within each processing element can be defined arbitrarily with the restriction that it has to be completely local, that is, it must depend only on the current values of the input signals arriving at the processing element via impinging connections and on values stored in the processing element's local memory.*

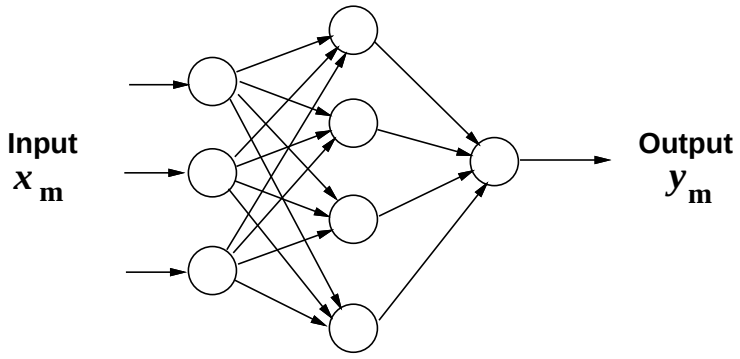


Figure 1.2: General neural network architecture

A basic kind of feed-forward neural network architecture is shown in Fig.1.2. There is an input layer, output layer and one or more number of hidden layers connected to each other by set of weights. We define the training set $\{x_m, y_m\}_{m=1}^{M}$ as the set of inputs x_m and the corresponding desired or optimal outputs y_m used to train the network. Learning in these kinds of system is carried out, typically, by changing (adjusting) the connection weights in response to the environment and the training data. A more detailed analysis of the neural architecture and hardware including

©Sethu Vijayakumar 1998

discussion on the basis functions (function specifying the input-output relationship of the hidden layer neurons) will be provided in Section 2.4. It is to be noted that although we use a neural network (with a particular architecture), the theory developed here can be studied independent of the learning system to a large extent since we will be generalizing the learning in these systems as an inverse problem of non-linear function approximation.

1.4.3 The present and the shortcomings

We will discuss the computational learning theory by using the neural network learning as the base line comparison tool. A classical framework for this problem is the *approximation theory* since all approximation schemes can be mapped into some kind of network that can be dubbed as "neural network". Related fields are *system identification techniques* (where it is possible to choose the input set) and *system estimation techniques* (when input-output pairs are given). For the sake of explanation, we consider the optimal approximation f_0 to be selected from a set of parametrised functions $f(x;w)$. In general, in approximation theory, three distinct stages can be identified [PG90].

1. the problem of which approximation class to use, i.e., which classes of function $f(x)$ can be effectively approximated with which set or class of approximating functions $f(x;w)$. This is the *representation* problem.

2. which criterion to use to find the optimal approximation $f(x;\hat{w})$ to the original function $f(x)$. This is the *learning* problem.

3. the problem of devising an efficient algorithm to do the approximation. This is the *implementation* problem.

So far, a lot of research work on this topic has been dealing with these three distinct stages, which ought to be treated individually, in a mixed up roundabout fashion. For example, in the popular Back Propagation algorithm which is a tool for searching within the non-linear solution space, the optimization criterion generally used in step 2 (from the above list) is of the Least Mean Squared type. Generalization ability of such learning is discussed in great detail inspite of the fact that the optimization criterion does not directly say anything about reducing errors outside of the training set; in other words there is no restriction on the learned function in areas of novel input. Smoothness constraints and restriction over weights (penalizing a factor of squared sum of connection weights) is often prescribed as regularization techniques to overcome effect of noise and reduce variance of the solution [KH92, Wil95, Ber86], i.e. in other words changing the optimality condition (Step 2). However, what is

©Sethu Vijayakumar 1998

actually being done is limiting the search space under which we are searching for the optimal solution by restricting the frequency content of the search space. The noise suppression and reduction of variance (at the cost of higher bias) is only a net effect.

Incremental learning, which is an important characteristic of biological learning systems is another area where a lot of research has been going on. Works on these lines by Zhang[Zha94], Jutten[JC95] and Dunkin[DSTK97] though being *incremental* in the nature of addition of training data or increase in the number of hidden units, almost always involves re-learning or re-training with the addition of training data or increase of hidden units. Other incremental approaches like Kadirkamanathan and Niranjan [KN93] and Yingwei [YSS97] have worked on specific radial basis function networks with sequential learning capability. However, in their work, the optimization criterion used reduces to an arbitrarily smoothed version of the memorization learning criterion, which may not be appropriate from a generalization point of view. There is a need to derive *exact* incremental learning techniques which can be extended to any type of optimization criteria and computes, without re-learning, the incremental learning solution that is exactly equal to the results of the batch optimization on the complete data set.

Moreover, there is a need to look into theoretical basis for the selection of optimal training data for maximal generalization capability. In systems where we have the freedom to sample at will, this result will be essential in reducing learning cost and improving system performance. Various active learning algorithms have been proposed based on stochastic assumptions and others based on maximizing information measures [Mac92] like the Kullback-Liebler divergence and Fisher's information measure. Most of these theoretical methods [Coh94, Whi89] suffer from the fact that, often, the information matrix is singular and these methods break down.

Finally, the most basic ingredients missing in incremental learning systems which perform in high dimensional learning tasks is the ability to learn incrementally in the presence of large number of inputs that are possibly redundant or irrelevant. In addition, the system should be able to incrementally allocate appropriate number of resources with more data since apriori knowledge in these are extremely limited. Learning has to proceed without interference, i.e., without forgetting useful knowledge acquired from previous data.

In the next section, we use the background built up so far along with the deficiencies pointed out in the current approaches to chalk out a set of objectives that we will be addressing and aiming to solve during the course of this dissertation.

©Sethu Vijayakumar 1998

1.5 The What and Why of this research work

As mentioned in the previous sections, we look at the problem of learning in artificial NNs as a function approximation problem. Under this framework, we use functional analysis as a tool for formalizing the various concepts of learning including model selection, search space, sampling and optimal reconstruction (or approximation). Previous research work on these topics have used this framework to find optimally generalizing solutions under various optimization criteria [Oga90, OO86, Oga92b, Oga86, deF90] and also bring to bear results from the mature fields of image processing and signal analysis into the area of learning theory [ON84, YO92, AH77]. Solutions for various optimization criterion and their generalized forms are provided in Section 2.3. The above mentioned framework clearly divides the *representation* problem, *learning* or optimization problem and the *implementation* problem in separate stages as will be described later. Moreover, functional analytic approaches have been proved to improve generalization ability because of their focus on the original function space rather than the sampled space [Oga92b].

However, when an optimal learning operator has been computed and an additional training data is made available, we need to incorporate this addition into the framework. The need for incremental learning also arises when we are dealing with real time systems in which the data comes in a continuous stream and we are required to make adjustments based on continually learned results. Re-learning using the entire training set in a possible but not very efficient solution; highly unacceptable in real time learning situations like the adaptive control of a robot arm. We should rather effectively use the results computed at the previous stage to improve the computational efficiency of the incremental learning task. To this effect we define the following objective:

1. Devise a scheme for efficiently incorporating a new training data and compute the new learning operators and learned function incrementally.

Based on this, intermediate results of training will be available, which can in turn be used to adapt or modify a sampling scheme based on newly available information.

We encounter cases in which we are allowed to choose the sampling scheme and probe the environment at desired location. Under the circumstances, it is essential that we select training data that is maximally informative and improves generalization. Hence, we define the next objective as:

2. Find the best sampling scheme (or best probing location) to improve the generalization and reduce uncertainty in the resulting solution.

This will provide a theoretical basis for optimal sampling; also referred to as *active learning* under the present learning framework.

The above mentioned global learning schemes (and almost all other exact incre-

©Sethu Vijayakumar 1998

mental schemes) do not scale very favorably with high complexity or dimensionality due to computational cost and numerical inaccuracies in approximation techniques. Moreover, they are very highly dependent on apriori knowledge of some extent for their success and it remains an open research issue how the size and structure of the learning schemes should be chosen such that their function approximation capabilities matches the learning problem, an issue dealt with under the bias-variance tradeoff [GBD92]. Hence, we want to devise a working incremental learning tool that has the following properties:

3. Devise a learning scheme which can learn incrementally in the absence of apriori knowledge, can scale well to high dimensional problems, can cope with redundant or irrelevant inputs, allocate resources incrementally and learn without interference.

This will be handled in the second part of the dissertation by using a nonparametric learning scheme which is heavily data driven and make very little commitments above the 'size' of the learning system.

All these learning techniques (and any other learning scheme) have the common problem of dealing with what kind of search space to look for solutions in. Some put in constraints like smoothness or regularization to restrict the search space while others deal with the frequency content of the search space. A systematic approach to discussing the problems associated with this will be dealt with under:

4. Discuss optimal model selection strategies for different kinds of problems and interpret standard techniques in the functional analytic framework.

The next section gives the format in which this thesis is arranged and points out which of the objectives discussed here are handled under which chapters.

1.6 Organization of this dissertation

The dissertation is essentially divided into two distinct but highly related and supplementary parts. The first Part, which includes Chapters 2 through 6, deal with exact methods of incremental and active learning. Part Two, encompassing Chapters 7, 8 and 9 deal with problems of incremental learning in high dimensional space in the absence (or in the presence of very little) apriori knowledge.

Chapter 2 introduces the functional analytic framework for learning in neural networks and discusses various optimization criteria and their solution. Methods of constructing Optimally Generalizing NN are discussed here. Chapter 3 deals with the problem of model selection and interprets standard strategies like regularization

©Sethu Vijayakumar 1998

under this framework. Chapter 4 provides closed form solutions for computationally efficient incremental learning using results on pseudo-inverses of positive definite matrices. These incremental results are tested in the learning of sensorimotor maps of a 2-Degree Of Freedom(DOF) robot arm in Chapter 5. Instead of using given training data, Chapter 6 deals with active learning for selecting maximally informative training sets using Wiener information measure as an example.

Chapter 7 looks at problems faced by global exact methods in high dimensional problems in the absence of apriori knowledge. Analysis of real motion data in humans and robot reveal local low dimensional structure and methods to exploit these locally such as PCA, Factor Analysis etc. are discussed here. Chapter 8 uses dimensionality reduction techniques evaluated in the previous chapter with a locally weighted non-parameteric regression scheme to develop the Local Adaptive SubSpace regression (LASS) algorithm, a receptive field based learning system that works in high dimensional spaces with very little structure. Chapter 9 evaluates the effectiveness of the LASS algorithm in artificial examples, learning in real 7-DOF anthropomorphic robot arm and in working with real human motion data.

Finally, Chapters 10 and 11 summarize the contributions of this work and lists out a set of points that will be the topic of future research along with new ideas that came up and could be pursued as a different line of thought.

©Sethu Vijayakumar 1998

©Sethu Vijayakumar 1998

Part II

Exact Incremental and Active Learning

2

Functional Analytic Approach to Neural Network Learning

2.1 Introduction

Conventionally, learning in neural networks is carried out by changing the weights in the connection of the network through some heuristic and hence, realizing the desired function f. However, here we will use a functional analytic method for formalizing the concept of learning and provide analytic solutions in the linear optimization domain using approximation theory concepts. Functional analytic approach to learning in Neural Networks(NNs) has been proved to improve generalizing ability because of their focus on the original function space rather than the sampled space [Oga90, Oga92b, OO86]. In Section 2.2, we will use functional analysis as a tool to formalize the concept of learning in NNs and define concepts of search space, sampling location, sampling operator and the reconstruction operator in the above framework. It had been mentioned in the Part I that it is essential to clearly define what is being aimed for and what are the errors being minimized before we look into how to obtain optimal solutions. To this effect, Section 2.3 considers various optimization criteria and their optimal solutions along with their interpretations. Finally, the correspondence between conventional weight space optimization and functional space optimization followed here is clarified and the method of selecting optimal number and type of basis functions along with optimal values for the connection weights for hardware realization of the Optimally Generalizing Neural Network (OGNN) is provided in Section 2.4.

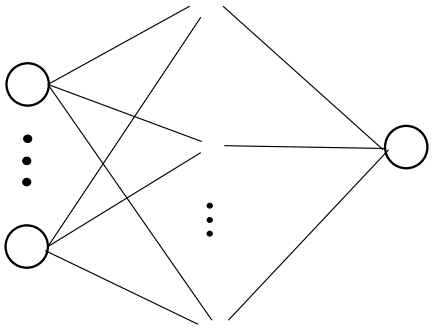


Figure 2.1: NN as a real valued function

2.2 Learning as an inverse problem

Let us consider a three-layer feedforward neural network whose number of input, hidden, and output units are, without loss of generality, L, N, and 1, respectively as shown in Fig.2.1. Let $\xi_1, \xi_2, ..., \xi_L$ be the inputs to the input layer neurons. Let these inputs constitute the elements of an L-dimensional vector x, referred to as the 'input vector'. An input-output relation of the n-th hidden unit is denoted by $u_n(x)$ and called the basis function. In conventional NNs, the following basis function is used:

$$u_n(x) = \sigma(\sum_{l=1}^{L} w_{nl}\xi_l), \tag{2.1}$$

where $\sigma(.)$ is a sigmoidal function. However, here we use more general basis functions of L-variables. Let w_n be the corresponding weight connecting the n-th hidden unit and the output unit. In this case, the network can be considered as a real valued function $f_0(x)$ of L variables and expressed as follows:

$$f_0(x) = \sum_{n=1}^{N} w_n u_n(x). \tag{2.2}$$

We define some notations relating to training data as used here.

$\{x_m\}_{m=1}^{M}$: A *training set* given as a set of M input vectors.

$\{y_m\}_{m=1}^{M}$: The corresponding desired output values, where $y_m = f(x_m)$.

y: Vector consisting of elements $\{y_m\}_{m=1}^{M}$.

$\{x_m, y_m\}_{m=1}^{M}$: Set of training data.

©Sethu Vijayakumar 1998

Definition 2.1 (Learning Problem) *The learning problem is to construct a neural network by using a training set so that the NN expresses the best approximation $f_0(x)$ to a desired function $f(x)$ under some given learning criterion.*

The learning problem of neural networks can now be discussed from the point of view of function approximation. In other words, the learning problem can be regarded as the same problem as that of the general sampling theorem [Oga86] and image restoration [ON84, AH77]. In this context, x_m and y_m can be called the m-th *sample point* and the corresponding *sample value* of the desired function f, respectively.

Let H be the set of all functions f to be approximated by the neural networks. Assume that H is a Hilbert space[1] with a *reproducing kernel* $K(x,x')$. Let D be the domain of the basis functions, which is a subset of the L-dimensional Euclidean space R^L. The reproducing kernel $K(x,x')$ is a bivariate function defined on $D \times D$ which satisfies the following two conditions:

1. For any fixed x' in D , $K(x,x')$ is a function in H.
2. For any function f in H and for any x' in D, it holds that

$$(f, K(.,x')) = f(x'), \tag{2.3}$$

where the left hand side of eq.(2.3) denotes the inner product in H.

In the theory of Hilbert space, arguments are developed by regarding a function as a point in that space. Thus, things such as 'value of a function at a point' cannot be discussed under the general framework of Hilbert space. However, if the Hilbert space has a reproducing kernel, then it is possible to deal with the value of a function at a point. Indeed, if we define functions $\psi_m(x)$ as

$$\psi_m(x) = K(x, x_m) : 1 \leq m \leq M, \tag{2.4}$$

then, the value of f at a sample point x_m is expressed in Hilbert space language as the inner product of f and ψ_m as

$$f(x_m) = (f, \psi_m). \tag{2.5}$$

Once the training set $\{x_m\}_{m=1}^{M}$ is fixed, the vector y is uniquely determined from f. So, we can introduce an operator A which transforms f to y:

$$y = Af. \tag{2.6}$$

[1] Chapter 3 will deal in more detail about problems of model selection and how to choose a particular Hilbert function space for particular kinds of problems and how to construct reproducing kernels for them.

©Sethu Vijayakumar 1998

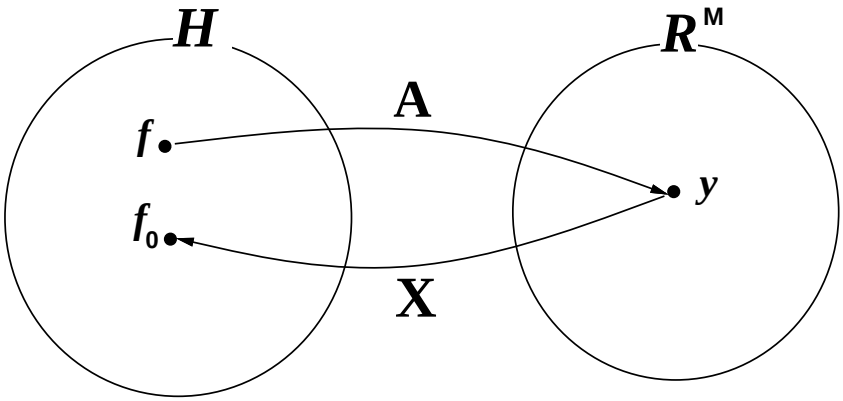


Figure 2.2: NN learning as an inverse problem

The operator A, called the *sampling operator*, becomes a linear operator even when we are concerned with nonlinear neural networks. It is expressed by using the Schatten product as

$$A = \sum_{m=1}^{M} e_m \otimes \overline{\psi_m}, \tag{2.7}$$

where $\{e_m\}_{m=1}^{M}$ is the so-called natural basis in R^M, i.e., e_m is the M-dimensional vector consisting of zero elements except the element m equal to 1. The Schatten product denoted by $(. \otimes \bar{.})$ is defined by

$$(e_m \otimes \overline{\psi_m})f = (f, \psi_m)e_m. \tag{2.8}$$

Hence, the learning problem is the problem of obtaining an estimate , say f_0, to f from y in the model (See Fig.2.2). This can be considered as an *inverse problem* equivalent to obtaining an operator X which provides f_0 from y:

$$f_0 = Xy. \tag{2.9}$$

The operator X is called the *learning operator*. It is also referred to elsewhere in other fields by different names like *restoration operator, reconstruction operator, approximation operator*, etc. This operator X can be optimized based on different criteria. These criteria, known as the *learning criteria*, are of different types like the memorization criterion, the Wiener criterion, the projection criterion, etc. [OO86]. The next section will deal in greater detail about the various optimization criteria and their optimal solutions.

©Sethu Vijayakumar 1998

2.3 Generalization ability and optimization criterion

2.3.1 Generalization ability

Learning with multilayer feedforward neural networks has been an attractive proposition due to their universal function approximation capabilities with sufficient number of hidden units. One of the most desirable characteristics of such networks is their ability to learn from examples and to generalize from the training set to similar data not contained in the training examples.

Definition 2.2 (Generalization ability) *Generalizing ability of a neural network is broadly defined as the ability of the NN to provide correct outputs, not only for the set of training data* $\{x_m, y_m\}_{m=1}^{M}$*, but also for the novel inputs which comes from the untrained region of the original mapping or function.*

Generalization error, as has been described in Chapter 1, can be interpreted in functional analytic framework as the squared error norm outside the training set; for square-integrable functions, it is a finite sample approximation of eq.(1.2)

$$E_G = \int_D [f(x) - f_0(x)]^2 dx, \tag{2.10}$$

where D is the domain of f in the input space. The generalizing ability of the network basically depends on the optimization criterion chosen to measure the closeness between the learned function and the original function. For example, if the criterion checks for the closeness in the sampled space alone and not in the original function space, then, we cannot say anything about the novel samples, i.e., points outside the training set, in general. Hence, the generalization ability is not guaranteed. On the other hand, closeness in the original function space can also have different meanings like closeness with respect to a particular function or in an averaged sense with respect to an ensemble of functions.

As mentioned in Section 2.2, a variety of criteria have been used to measure the closeness between the original desired function and the learned function. In practice, a suitable criterion should be chosen based on the requirements of the problem. Examples of some criteria and their significance, shortcomings and interpretations have been provided in the following subsections.

2.3.2 Memorization criterion

An example of criterion which reduces error only with respect to the training data is the memorization learning. It is also referred to as the Least Mean Squares (LMS)

©Sethu Vijayakumar 1998

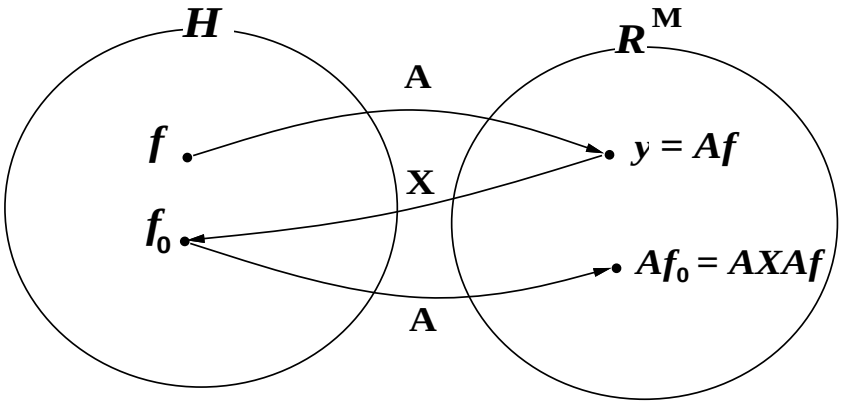


Figure 2.3: Memorization learning : Optimization in the sampled space

criterion in the field of statistical approximation theory. The formal definition of the criterion is as given below.

Definition 2.3 (Memorization criterion J_M) *For all $y = Af$, minimize the following functional with respect to X:*

$$J_M[X] = \sum_{m=1}^{M} \{f_0(x_m) - y_m\}^2 = \|AXy - y\|^2, \tag{2.11}$$

where $\|.\|$ is the norm in the sampled space R^M.

The popular Back Propagation (BP) algorithm (Werbos 1974; Parker 1982; and Rumelhart, Hinton and Williams 1986) is an algorithm designed to search for optimal reconstruction operator X in the non-linear domain under the memorization optimization criterion. The graphical representation of the errors being considered in the sampled space R^M is given in Fig.2.3. This learning criterion J_M minimizes the error, in other words, asks for correct responses for the training inputs, but does not specify anything about the novel data or untrained data. Therefore, any algorithm based on J_M does not guarantee the generalization ability of the neural network in general. In this sense, we call J_M learning memorization learning. The suffix 'M' of J_M refers to 'memorization'.

Lemma 2.1 *The necessary and sufficient condition for the learning operator X to satisfy memorization learning is that X is a 1-inverse of the sampling operator A. In other words, X satisfies the equation*

$$AXA = A. \tag{2.12}$$

©Sethu Vijayakumar 1998

Corollary 2.1 (General solution for memorization criterion) *A general form of the memorization learning operator X is given as*

$$X = A^{\dagger} + Y - A^{\dagger}AYAA^{\dagger}, \tag{2.13}$$

where Y is any operator from R^M to H and † represents the Moore-Penrose generalized inverse[2].

In the noiseless case, it is possible to make the LMS error described by eq.(2.11) zero. This means that the reconstructed function f_0 will have values $y_m = f(x_m)$ at the training data points x_m, i.e., $Af_0 = y$. Then, the following theorem holds.

Theorem 2.1 *Any reconstructed function f_0 such that*

$$f_0(x_m) = f(x_m) = y_m, \quad m = 1, ..., M \tag{2.14}$$

has the form

$$f_0 = f_0^{(1)} + f_0^{(2)} = A^{\dagger}y + (I - A^{\dagger}A)f' \tag{2.15}$$

where f' is an arbitrary function in the original function space H, $f_0^{(1)} = A^{\dagger}y$ and $f_0^{(2)} = (I - A^{\dagger}A)f'$. The functions $f_0^{(1)}$ and $f_0^{(2)}$ are orthogonal to each other and satisfy the condition

$$f_0^{(1)}(x_m) = y_m, \;\; f_0^{(2)}(x_m) = 0, \quad m = 1, ..., M. \tag{2.16}$$

Moreover, the minimum norm reconstructed function is the linearly reconstructed function $f_0^{(1)}$.

The result given in eq.(2.16) has an important interpretation. The unique minimum norm solution, $f_0^{(1)} = A^{\dagger}y$, a special case of the 1-inverse solution of memorization learning, has the same values $y_1, ..., y_M$ at the sample points $x_1, ..., x_M$ as the original function f. The extra term $f_0^{(2)} = (I - A^{\dagger}A)f'$ has zero values at these points but it will affect the reconstructed function elsewhere. By learning with the memorization criterion (LMS criterion), it is *impossible* to control the function $f_0^{(2)}$ since eq.(2.15) is the optimal solution for any f'. Hence, there is no guarantee that f_0 would be close to f outside the training data. There exists only heuristic remedies for controlling the fluctuation of the learned function using BP algorithm outside the training points like regularization or early stopping of training etc; However, it has to be understood that this is not the drawback of the BP algorithm; rather it is the problem of the LMS criterion.

Minimizing the generalizing error described in eq.(2.10) directly instead of the training set error, as in the memorization criterion, is essential while theorizing about generalizing ability. We will look at some optimization criterion which attempts this in the next section.

[2] Refer to Section 4.3 for a detailed description of the Moore-Penrose generalized inverse

©Sethu Vijayakumar 1998

2.3.3 Wiener criterion

The formal definition for the Wiener criterion is given below.

Definition 2.4 (Wiener criterion J_W) *Minimize the following functional with respect to X:*

$$J_W[X] = E_f \|XAf - f\|^2, \tag{2.17}$$

where $\|.\|$ is the norm in the original function space H and E_f is the expectation taken over the ensemble $\{f\}$.

An operator X satisfying the above criterion is called a Wiener learning operator. This criterion is often used in the field of image restoration[AH77]. The criterion aims at reducing the difference between the original function f and the function f_0 reconstructed by using the learning operator X. The thing to be noted is that this minimization is done in the original function space H and in an *averaged sense* with respect to all the possible functions to be approximated.

Here, we will restrict ourselves to only the linear solutions to the minimization criterion. Hence, we will consider only linear learning operators X as the possible solutions. However, limiting the learning operator X to a linear one does not imply that the model can represent or realize only linear functions f_0. A linear learning operator X can be used to realize a non-linear function f_0 in our learning model.

The Wiener criterion can be transformed into a more useful form. Let R be the *correlation operator* of all the functions to be approximated and be defined as

$$R \equiv E_f(f \otimes \bar{f}). \tag{2.18}$$

Lemma 2.2 *[YO92] A necessary and sufficient condition for the learning operator X to satisfy the Wiener criterion is that the following equation is satisfied.*

$$XARA^* = RA^*. \tag{2.19}$$

where A^* is the adjoint operator of A.

The eq.(2.19) can be interpreted in the following way. All learning operators satisfying the Wiener criterion can reconstruct the functions in the range $\mathcal{R}(RA^*)$ completely from the sample vector $y = Af$, i.e.,

$$Xy = f, \text{ for } f \in \mathcal{R}(RA^*). \tag{2.20}$$

Corollary 2.2 (General solution for Wiener criterion) *A general form of the Wiener learning operator X (solution of eq.(2.19)) is given as*

$$X = RA^*U^\dagger + Y(I - UU^\dagger), \tag{2.21}$$

©Sethu Vijayakumar 1998

where

$$U = ARA^*. \tag{2.22}$$

Here, Y is any operator from R^M to H.

Once the training set is fixed, A can be calculated using eq.(2.7), and hence, corresponding to this sampling operator, a learning operator X satisfying the Wiener criterion can be obtained using eqs.(2.21) and (2.22).

Corollary 2.3 (Learned function using Wiener criterion) *The function f_0 learned using the Wiener criterion can be written, using the learning operator X of eq.(2.21) as*

$$f_0 = Xy = RA^*U^\dagger y. \tag{2.23}$$

We will be using the Wiener learning criterion as an example for implementing our theory on incremental learning [VO95] and deriving results on incremental computation of the learning operator and the learned function. A more detailed analysis of the properties of the Wiener learning will be carried out in Section 4.5, where we will consider the dynamic properties with addition of training data.

2.3.4 Projection criterion

The memorization criterion J_M does not guarantee the generalization ability of a neural network. The Wiener criterion J_W provides the best approximation in the averaged sense with respect to the ensemble $\{f\}$. It does not imply the best approximation to any individual f. It may be sufficient for some cases. However, there are situations where the best approximation f_0 to an individual f is needed. It is often said that we cannot obtain such an f_0 because we do not know the original function f. However, this is not true. The projection criterion J_P provides an answer to the problem.

Definition 2.5 (Projection criterion J_P) *For all $f \in H$, determine X such that*

$$f_0 = XAf \tag{2.24}$$

becomes the best approximation to the original function f in the subspace $\mathcal{R}(A^*)$.

The learned function $f_0 = XAf$ is always contained within $\mathcal{R}(X)$. Hence, while evaluating the nearest or the best approximation to f, we have to select the answer from within this subspace. When we use a linear operator X for learning f, the range of X becomes a closed subspace of H. Hence, the closest approximation of

©Sethu Vijayakumar 1998

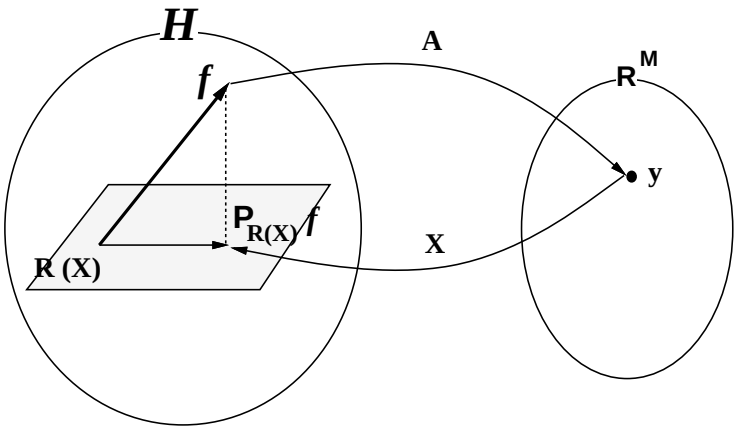


Figure 2.4: Projection learning

f in $\mathcal{R}(X)$, as shown in Fig.2.4, is an orthogonal projection of f onto $\mathcal{R}(X)$. The following lemma shows that we can obtain the orthogonal projection of f onto $\mathcal{R}(X)$ from y even when we do not know f.

Lemma 2.3 *[Oga81, ON84] Let S be a closed subspace of H. For any function f of H, its orthogonal projection onto S is computable from $y = Af$ if and only if S is included in the subspace $\mathcal{R}(A^*)$.*

Based on this lemma, the condition for being able to reconstruct the function completely becomes

$$\mathcal{R}(X) \subseteq \mathcal{R}(A^*). \tag{2.25}$$

Since we are determining the orthogonal projection onto the subspace $\mathcal{R}(X)$, it is better that for optimal results, this subspace be as large as possible. Hence, based on eq.(2.25), we will concentrate our attention on the projection onto the subspace $\mathcal{R}(A^*)$. Hence, we can re-write the projection criterion as determining an X such that it satisfies

$$XAf = P_{R(A^*)}f : f \in H \tag{2.26}$$

It is for this reason that in Definition 2.5, the learned function becomes the best approximation in the *subspace* $\mathcal{R}(A^*)$. This operator X is called a projection filter in the field of image restoration[ON84].

Corollary 2.4 (General solution for projection criterion) *A general form of the projection learning operator X (solution of $XA = P_{R(A^*)}$) for the* noiseless *case is given as*

$$X = A^\dagger + Y(I - AA^\dagger), \tag{2.27}$$

where Y is any operator from R^M to H.

©Sethu Vijayakumar 1998

Corollary 2.5 (Learned function using projection criterion) *The function f_0 learned using the projection criterion can be written, using the learning operator X of eq.(2.27) as*

$$f_0 = Xy = A^{\dagger}y. \tag{2.28}$$

As mentioned earlier in Section 2.3.1, each of the optimization criteria has it's own characteristics with regard to what kind of error it optimizes and what space it searches for solutions in. The criterion to be used should, hence, be chosen based on the type of problem and objective to be achieved. In the next section, we will proceed to show the analogy between conventional weight modification based neural networks and the present framework used here and then, go on to explain how to realize an optimally generalizing neural network (OGNN) architecture based on the optimal learning operators derived in this section.

2.4 Realization of optimally generalizing NN (OGNN)

First we will describe the standard architecture used in multilayer feedforward networks. A simple model of an L-input, 1-output and N-hidden units neural network is shown in Fig.2.5. As mentioned in Section 2.1, the conventional neural networks learns through modification of weights in the connections.

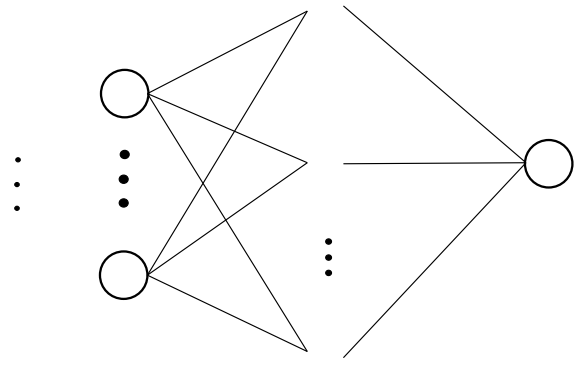


Figure 2.5: Conventionally used NN architecture

The summation of the inputs, each weighted by it's corresponding connecting weight w_{nl}, acts as the input to the hidden layers. This value is then processed using the activation function of the hidden unit, which is usually a sigmoidal function. This

©Sethu Vijayakumar 1998

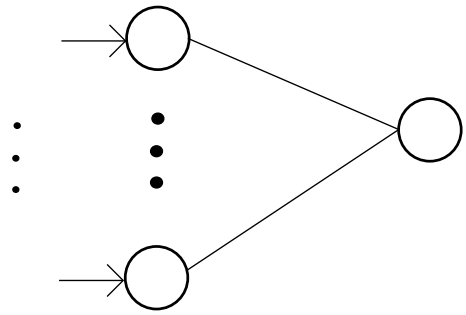


Figure 2.6: nth hidden unit of the neural network

network learns by changing the weight sets w_{nl}: between the input layer and the hidden units and w_n: between the hidden units and the output unit. In general, neural networks of the type shown in Fig.2.5 is used widely.

Based on the weights w_{nl}, we can assume that the hidden layer is in fact realizing an input-output or basis function, denoted by

$$u_n(x) = \sigma(\sum_{l=1}^{L} w_{nl}\xi_l). \tag{2.29}$$

This relationship is graphically depicted in Fig.2.6. This implies that by changing the weight set $\{w_{nl}\}_{l=1}^{L}{}_{n=1}^{N}$, we also change the basis function u_n of the hidden unit. Hence, changing the weights $\{w_{nl}\}_{l=1}^{L}{}_{n=1}^{N}$ using learning algorithms in the above kind of networks corresponds to fixing the input-output relationship of the hidden units.

By adjusting both the weight sets $\{w_{nl}\}_{l=1}^{L}{}_{n=1}^{N}$ and $\{w_n\}_{n=1}^{N}$ using the training data $\{x_m, y_m\}_{m=1}^{M}$ based on some fixed learning algorithm like the back-propagation algorithm etc., an LMS error criterion minimization leads to realization of the function f_0.

The model of neural network used in this dissertation, although sharing the fact that they are also multilayer feedforward networks, realize a more general form of the basis functions as will be clear in the following analysis. As is evident from the Fig.2.7 which represents the architecture used here, there are no weights in the connections between the input layer and the hidden layer. So, corresponding to the problem of changing the weights $\{w_{nl}\}_{l=1}^{L}{}_{n=1}^{N}$ in the conventional NN model shown in Fig.2.5, we have the problem of selecting the basis functions u_n itself (refer eq.(2.29)). This ensures that the activation function need not necessarily be a sigmoidal activation function. The method of selection of the basis functions themselves will be dealt

©Sethu Vijayakumar 1998

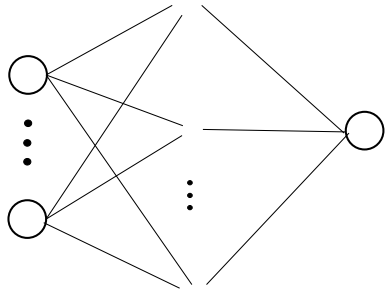


Figure 2.7: NN architecture used in this dissertation

with later in this section.

In Section 2.3, we have seen how to derive optimal learning operator X analytically once we are given the set of training data and the optimization criterion. However, when we look at the analogy with conventional NNs, from an engineering point of view, we also have to realize this optimal function in hardware, in other words, there is the problem of deciding what architecture to choose, the kind and number of basis functions and how to implement the connection weights to realize the optimally generalizing solution. In general, we will call such networks Optimally Generalizing Neural Networks(OGNN). The realization of these networks can be done by using the following lemma.

Lemma 2.4 *[Oga95] A practical implementation of the Optimally Generalizing Neural Network (OGNN) can be realized by determining the number of hidden units N, the basis functions of the hidden units $\{u_n\}_{n=1}^N$ and the connection weights $\{w_n\}_{n=1}^N$ (refer Fig.2.7) as follows:*

1. *Fix N such that*
$$N \geq dim(\mathcal{R}(XA)) = rank(XA). \tag{2.30}$$

2. *Fix $\{u_n\}_{n=1}^N$ such that, $\mathcal{L}(\{u_n\}_{n=1}^N)$, the subspace spanned by $\{u_n\}_{n=1}^N$ satisfy*
$$\mathcal{L}(\{u_n\}_{n=1}^N) \supset \mathcal{R}(XA). \tag{2.31}$$

3. *Compute the vector of weights $w \equiv (w_1 w_2 ... w_N)^T$ as*
$$w = Wy, \tag{2.32}$$
where W, an $N \times M$ matrix, is defined as
$$W = (T^*)^\dagger X + (I - TT^\dagger)Y. \tag{2.33}$$

©Sethu Vijayakumar 1998

Here, T is given by

$$T = \sum_{n=1}^{N} (e_n \otimes \overline{u_n}) \tag{2.34}$$

and Y is any operator from R^M to R^N.

The realization of the optimally learned function from the optimal weight vector w is obtained using the adjoint of T as $f_0 = T^* w$. The relationship between different operators, the optimal weights, the sample vector y and the optimal reconstructed function f_0 is shown graphically in Fig.2.8.

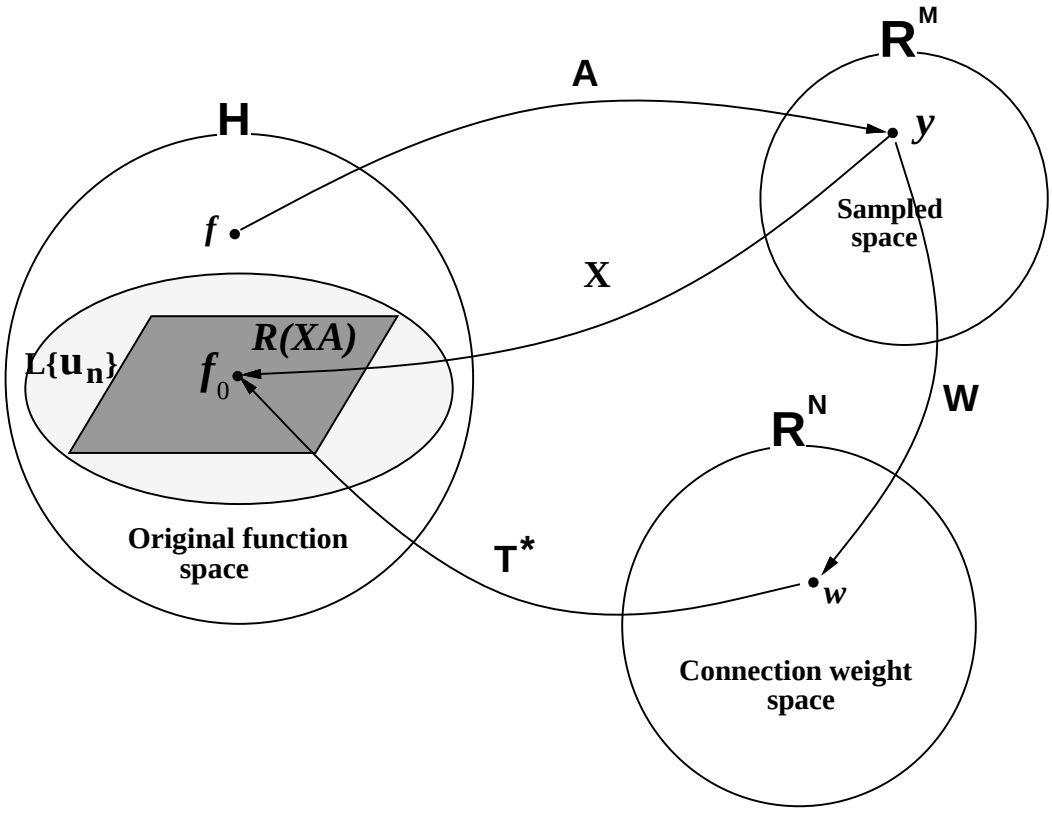


Figure 2.8: Functional analytic approach to NN learning.

Hence, using the above lemma, it is possible to realize the neural network through physical weights in one step after the learning operator X has been mathematically determined. Based on Lemma 2.4, we can also easily derive an algorithm for one cycle learning of the optimal weights as shown in the following corollary.

Corollary 2.6 (One cycle learning) *[Oga95] Let us assume that we select the number and type of basis functions u_n such that conditions (1) and (2) in Lemma 2.4 hold. Also, let Δ_m represent the m-th column of the matrix W given in eq.(2.33).*

©Sethu Vijayakumar 1998

If we determine $w(m)$ as

$$w(0) = 0 \tag{2.35}$$

$$w(m) = w(m-1) + y_m \Delta_m : m = 1, 2, ..., M \tag{2.36}$$

then, $w(M)$ coincides with the optimal weights w given in eq.(2.32).

Looking at the correspondence between the conventional NN model and the functional analytic model used in this dissertation, we see that

- Determining the weights $\{w_{nl}\}_{l=1}^{L}{}_{n=1}^{N}$ in Fig.2.5 corresponds to fixing the basis functions $\{u_n\}_{n=1}^{N}$ in our model. This also means that at this stage, the transformation T is defined using eq.(2.34).
- Determining the weights $\{w_n\}_{n=1}^{N}$ in Fig.2.5 corresponds to determining the operator W in Fig.2.8 which maps the sample vector y to the weights.

Note: In the conventional BP approach the two sets of weights $\{w_{nl}\}_{l=1}^{L}{}_{n=1}^{N}$ and $\{w_n\}_{n=1}^{N}$ are altered based on the algorithm without any apparent difference between the two sets. However, using the model used in this dissertation, we can clearly identify the difference between the two weights. The weight set $\{w_{nl}\}_{l=1}^{L}{}_{n=1}^{N}$ corresponds to determining the range of the reproducible or learnable functions based on the *sampling scheme* alone; which corresponds to determining the basis functions $\{u_n\}_{n=1}^{N}$ in our model. Therefore, this uses only the *training set* and not the corresponding *sample values*. Evaluation of the weight set $\{w_n\}_{n=1}^{N}$ corresponds to fixing the learned function f_0 within the range $\mathcal{R}(XA)$, which actually depends on the sample values $\{y_m\}_{m=1}^{M}$.

©Sethu Vijayakumar 1998

©Sethu Vijayakumar 1998

3

Model Selection

3.1 Introduction

In approximation theory, we clearly see three stages in the development of the problem [PG90]. First, there is the *model selection* problem, which addresses which space to search for the optimal approximation. This is equivalent to the problem of selecting the architecture, the number of hidden units and basis functions in conventional NN literature. In our case, this is analogous to the question of which Hilbert space is relevant for a particular class of problems. The other two stages involves finding the *optimal approximation* within this model space and devising fast and cheap *implementations* of the algorithm. Many recent works on learning theory have solely concentrated on the merits and suitability of various model selection strategies in different real world learning problems [SUF97]. Although the emphasis of this dissertation is not on the model selection problem, in this chapter, we show that this framework is already well adapted for carrying out model selection and introducing bias in a *systematic* way. The problems associated with bias-variance is discussed and illustrated very briefly followed by a detailed description of how to use the functional analytic framework for dealing with the model selection problem. A variety of function spaces and methods of constructing their reproducing kernels is provided here.

3.2 Standard model selection strategies

When learning a function $f : H \rightarrow R^M$ from random training examples $\{x_m, y_m\}_{m=1}^M$, there is a well known tradeoff between the size of the training sample and the complexity of the function class being considered : If the class is too complex for the sample size, there is the risk of 'overfitting' the training data and resulting in an approximation which performs poorly on future test data. On the other hand, an overly restricted class can prevent us from considering any good candidates. The most common strategy for dealing with this dilemma is some form of automatic *model selection*, such as complexity penalization or repeated hold-out testing, to balance the tradeoff between complexity and data-fits.

Under the simplest formulation of model selection, the idea is to first stratify the hypothesis (or search space) H into a sequence of nested subclasses $H_0 \subset H_1 \subset ... = H$ and then (somehow) choose a class which has the appropriate complexity for a given training set. To understand how we might make this choice, note that for the given training set $\{x_m, y_m\}_{m=1}^M$, we obtain a corresponding sequence of empirically optimal functions, $f_0^* \in H_0$, $f_1^* \in H_1$ etc. that achieve the minimum average error with respect to some error criterion $E(f_i^*) = err(f, f_i^*)$. Although we can, in general, search for the optimal function with any kind of optimization criterion, the error itself is generally non-tractable at points other than the training points. Hence, an empirical error function $E(f_i^*) = \sum_{m=1}^M (y_m - f_i^*(x_m))$ is used as an error criterion. Note, however, that these errors are monotonically decreasing and therefore, choosing the function with the minimum empirical error simply implies choosing a function from the largest class. Therefore, the trick is to invoke some other criteria beyond empirical error minimization to make this choice.

Currently, two basic model selection strategies predominate. The most common strategy is *complexity penalization*. Here, we assign increasing complexity values $c_0, c_1, ...$ to the successive function classes, and then, choose the approximation from $f_0^*, f_1^*, ...$ that minimizes some prior combination of complexity and empirical error e.g. the additive combination $c_i + \lambda E(f_i^*)$. There are many variants to this basic approach, including the minimum description length principle of Rissanen[Ris86], "Bayesian" maximum a posteriori selection , structural risk minimization of Vapnik[Vap96], "generalized" cross validation of Craven and Wahba [CW79] and regularization [Moo92]. These strategies differ in the specific complexity values they assign and the particular trade-off function they optimize, but the basic idea is the same.

The other most common strategy is *hold-out testing*. Here, we ask, for the given set of training data, which function class H_i generalizes the best? We answer this by partitioning the training set $1, ..., M$ into a pseudo training set $1, ..., k$ and a hold-

©Sethu Vijayakumar 1998

out test set $k+1, ..., M$ and then, use the pseudo-training set to obtain a sequence of approximating functions(based on some optimization criterion). The optimal function class H_i is chosen to be that class for which the approximating function gives minimum error on the test set. Once, H_i has been selected, we compute the function $f_i^* \in H_i$ that optimizes some error criterion based on the *entire* training set. Again, there are many variants to the basic strategy. For example, there is the method of pseudo-train pseudo-test split many times and averaging the results to choose the final search (hypothesis) space; that is 10-fold cross validation, leave one out testing, bootstrapping etc; [Efr79, WK91].

The abundance of model selection strategies and different approaches to the problem raises the question of which techniques are best and when. Here, we will not directly attempt to answer this since this is beyond the scope of the current research. However, an understanding of the concept in the context of regression can be obtained by appealing to the standard bias/variance decomposition of the generalization error [GBD92], which we will briefly describe in the next section.

3.3 The bias-variance dilemma

One way to assess the suitability of a particular search (hypothesis) space H is to consider the *expected* (true) error that results from minimizing some optimization functional on H.

Let us define some notations used in this context.

- $f(x)$: true function
- $D \equiv$ {data sets of size M drawn from a given input distribution}
- $f^*(x, D_i)$: Optimal estimate (approximation) using a particular data set D_i from the set D.

The *expected* (true) error or generalization error can be written as:

$$GenErr(f^*) = \int \underset{D_i \in D}{E}[(f(x) - f^*(x, D_i))^2]p(x)dx. \tag{3.1}$$

by taking the average over all possible similar sized data sets and integrated over the range of input data. Here, $E[.]$ represents the average with respect to a particular distribution. This error can be decomposed into *bias* or *variance* terms by expanding around the *mean* approximated function $\underset{D_i \in D}{E}[f^*(x, D_i)]$ as follows[GBD92].

©Sethu Vijayakumar 1998

$$
\begin{aligned}
GenErr(f^*) &= \int (f(x) - \underset{D_i \in D}{E}[f^*(x, D_i)])^2 p(x) dx \qquad \text{“bias”} \\
&+ \int \underset{D_i \in D}{E}[(f^*(x, D_i) - \underset{D_i \in D}{E}[f^*(x, D_i)])^2] p(x) dx \quad \text{“variance”} \qquad (3.2)
\end{aligned}
$$

The first part is the squared deviation between the system and the expectation of the model over all possible training. It is called the squared *bias* as it measures the intrinsic bias the model has. The second part is the average of the squared fluctuations of the model around it's expectation; the *variance*. Accordingly, the generalization error can be separated into two components:

1. The squared bias induced by the choice of the model or hypothesis class.
2. The variance coming from the sampling.

If we consider a learning with a system which has a very broad range of possible solutions, then, we can be sure that in almost all the cases, the true function being modeled lies inside the search space of the learning system. This implies that we have a very small bias since *asymptotically* the system will reach the correct solution. However, the price to pay for small bias is high variance. A learning system sufficiently versatile to reasonably approximate a broad range of functions is necessarily sensitive to the idiosyncrasies of the particular data set. On the other hand, seriously restricting the class of possible functions will reduce variance drastically but suffer from the problem that the true function being modeled may not be contained in this restricted class. Ofcourse, if we can ensure the *right* bias, that is the best way to go. There is often a tradeoff between the bias and variance contributions to the estimation error, which makes fro a kind of “uncertainty principle”[Gre51].

The bias/variance dilemma can be circumvented if one is willing to give up generality, that is, *purposefully* introduce bias. In this way, variance can be eliminated or significantly reduced. Of course, one must ensure that the bias is harmless for the *problem under consideration*: the particular class of functions from which the estimator is to be drawn should still contain the regression function, or atleast a good approximation thereof. In essence, the bias needs to be *designed* for each particular problem.

From a statistical physics perspective, introducing bias may also be viewed as a means of decreasing an appropriately defined measure of entropy of the machine [CP87]. In the neurobiological sense, the biases correspond to information “internalized” during the course of phylogenetic and ontogenetic evolution. It may be a good idea for engineers solving problems, for example, in machine perception to look for inspiration in living brains and hope to discover the nature of pre-wired biases.

©Sethu Vijayakumar 1998

However, the hope that current connectionist networks inherit the biases of the real brain from the mere fact that they are built from "brain-like" processing elements seems farfetched [Cri89].

3.4 Using family of stratified Hilbert spaces

So far, we have seen how important it is to "look" in the right search space for the optimal answers. Too large an area, although assuring us of an ultimate correct solution, requires way too many training examples to be feasible. An incorrectly narrowed down search space provides *apparently* optimal answers with relatively small number of training data but suffers from the danger of missing out on the true function or even a good approximation of it. In developing the functional analytic framework of Chapter 2, we had started from a point by assuming that we search within a Hilbert space H.

What spaces are appropriate for what kind of problems is a very task dependent question. However, in this section, we give a formulation of Hilbert spaces and how to construct their reproducing kernels. We go on to provide examples of some Hilbert spaces which can be stratified (parametrised) into subspaces, which act as an appropriate basis for the model selection strategies discussed in the previous sections.

3.4.1 Hilbert space and reproducing kernel

Definition 3.1 (Inner product space) *An inner product in a (real or complex) vector space is a (respectively, real or complex) numerically valued function of the ordered pair of vectors x and y, such that*

1. $(f,g) = \overline{(g,f)}$,
2. $(\alpha_1 f_1 + \alpha_2 f_2, g) = \alpha_1(f_1,g) + \alpha_2(f_2,g)$,
3. $(f,f) \geq 0; (f,f) = 0$ *if and only if* $f = 0$.

An inner product space is a vector space with an inner product.

A *Hilbert space* is a *complete* inner product space; the definition of completeness can be obtained from any standard book on functional analysis.

Let us assume that a finite dimensional Hilbert space has an OrthoNormal Basis (ONB) given by $\{\varphi_n\}_{n=1}^{N}$. Then, the reproducing kernel of this function space

©Sethu Vijayakumar 1998

$K(x,x')$, a bi-variate function with properties defined in Section 2.2, which is necessary for implementing our functional analytic theory can be derived using the formulation

$$K(x,x') = \sum_{n=1}^{N} \varphi_n(x)\varphi_n(x'). \tag{3.3}$$

It is interesting to note that, for every Hilbert space, the reproducing kernel, if it exists, is unique [Sai97] and does not depend on the kind of basis functions used to construct it. We will call a Hilbert space with a reproducing kernel as a Reproducing Kernel Hilbert Space(RKHS). The next subsection shows examples of some useful Hilbert function spaces and method of constructing their reproducing kernels.

3.4.2 Some useful Hilbert spaces and their formulation

A function space that has extensive practical significance in one or multi dimensional signal processing is the space of *band-limited* functions, defined as a function whose Fourier transform is zero outside a hyper-rectangle in the frequency domain. Such a function space forms a Hilbert space (also referred to as the Paley-Wiener space) and it's reproducing kernel can be determined as follows. Consider for simplicity, the function of two variables; if the Fourier transform $F(\omega) = F(\omega_1,\omega_2)$ of the function $f(x) = f(\xi_1,\xi_2)$, defined as

$$F(\omega) = \int_{-\infty}^{\infty} f(x)e^{i\omega x}dx \tag{3.4}$$

is zero for $|\omega_1| > \sigma_1, |\omega_2| > \sigma_2$, then the sampling series is based on the functions

$$K(x) = \frac{\sigma_1\sigma_2}{\pi^2} sinc[\frac{\sigma_1}{\pi}\xi_1] sinc[\frac{\sigma_2}{\pi}\xi_2], \tag{3.5}$$

where $sinc$ is the function

$$sinc(u) = sin(\pi u)/\pi u. \tag{3.6}$$

The function $K(x,x') = K(x-x')$ forms the reproducing kernel for the Hilbert space of band-limited functions.

Fig.3.1 shows examples of such two-dimensional reproducing kernels for band-limited function spaces defined by parameters (a) σ_1=8, σ_2=8 and (b) σ_1=10, σ_2=3.

The approximation using these kinds of Reproducing Kernel Hilbert Spaces (RKHS) is close to the Radial Basis Function (RBF) networks but the kernel of eq.(3.5) or it's generalization to L dimensions,

$$K(x,x') = \prod_{l=1}^{L} \frac{\sigma_l}{\pi} sinc[\frac{\sigma_l}{\pi}(\xi_l - \xi_l')], \tag{3.7}$$

with $x = (\xi_1,...,\xi_l)$, $x' = (\xi_1',...,\xi_l')$, is not radially symmetrical in general. However, it has a central 'lump' whose width and shape is controlled automatically by the L

©Sethu Vijayakumar 1998

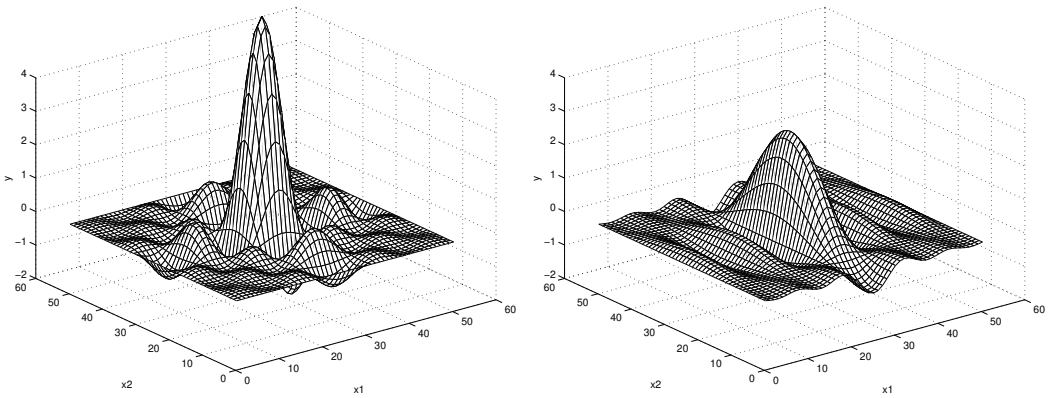


Figure 3.1: Reproducing kernels for 2-dimensional band-limited function spaces with cutoff frequency parameters (a) $\sigma_1 = 8, \sigma_2 = 8$ and (b) $\sigma_1 = 10, \sigma_2 = 3$. Note the radial asymmetry in (b).

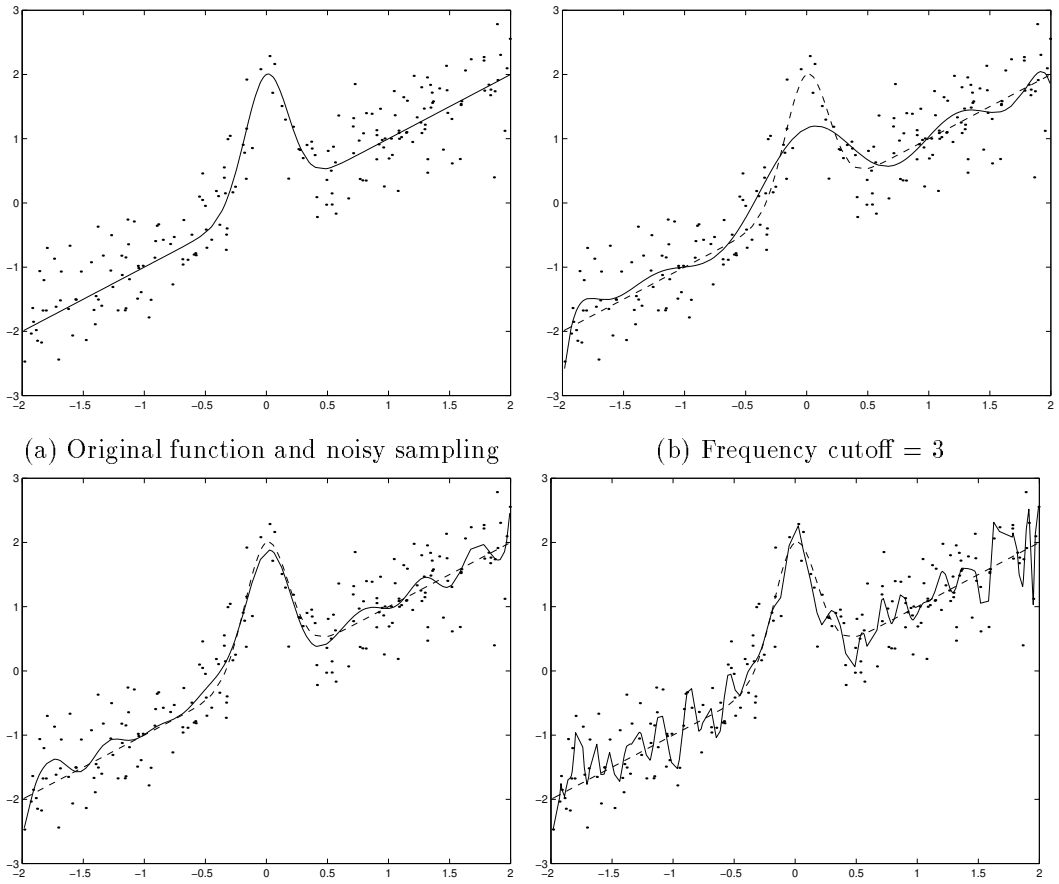
parameters σ_l specifying the band-limitation in the frequency domain. Note that the sampling theorem provides a proof that asymptotically, approximations using these kinds of basis functions can approximate any band-limited function to arbitrary accuracy. Adjustment of the parameters σ_l in eq.(3.7), which can potentially make this frequency band arbitrarily large, can be carried out through cross validation or other model selection techniques like complexity penalization in addition to systematically introducing prior biases on the smoothness by controlling the frequency content of the search space. A standard practice in NN literature of using regularizers e.g. in splines, may be looked upon as methods of controlling the frequency content of the search space, although their exact relationship is still a topic of active research.

Next, we show an example of approximation using the Paley-Wiener space. The original function to be approximated is a nearly linear function with a non-linear bump in the middle (See Fig.3.2(a)) given by

$$y = x + 2exp(-16x^2). \tag{3.8}$$

The function is sampled at 200 locations with an additive noise of the order of ten percent (Noise variance =0.1). The sampled locations are shown with dots in Fig.3.2. We show results of approximation using the projection learning criterion described in Section 2.3.4. Fig.3.2 (b), (c) and (d) show approximations (solid line) when we consider a band limited function space with band frequency cut-off $\sigma = 3$, $\sigma = 10$ and $\sigma = 40$, respectively. As can be seen from the comparison with the original function, function space with $\sigma = 3$ has a large bias, hence, the high frequency components of the functions are smoothed out. On the other hand, $\sigma = 40$ results in a approximation which has very low bias but highly depends on the noisy data; i.e. has high variance. A suitable tradeoff between bias and variance results when

©Sethu Vijayakumar 1998



(a) Original function and noisy sampling

(b) Frequency cutoff = 3

(c) Frequency cutoff = 10

(d) Frequency cutoff = 40

Figure 3.2: Function approximation using the projection learning criterion and using original function spaces with frequency cut-off (b) $\sigma = 3$, (c) $\sigma = 10$ and (d) $\sigma = 40$

we consider the function space with frequency cutoff $\sigma = 10$.

Another approach to selection of the relevant Hilbert space H involves parameterization of the space based on the highest degree of polynomial within the space. For example, if we consider the Gram-Schmidt orthogonalization of the independent polynomial functions $1, t, t^2, ...$ within the Hilbert space $L_2[-1, 1]$, where the inner product is defined as

$$(f, g) = \int_{-1}^{1} f(x)\overline{g(x)}dx, \tag{3.9}$$

we can generate a set of orthonormal sequences given by

$$e_n(t) = \sqrt{\frac{2n+1}{2}} P_n(t), \qquad n = 0, 1, 2, ... \tag{3.10}$$

where $P_n(t)$ are the well known Legendre polynomials

$$P_n(t) = \frac{1}{2^n n!} \frac{d^n}{dt^n}(t^2 - 1)^n. \tag{3.11}$$

©Sethu Vijayakumar 1998

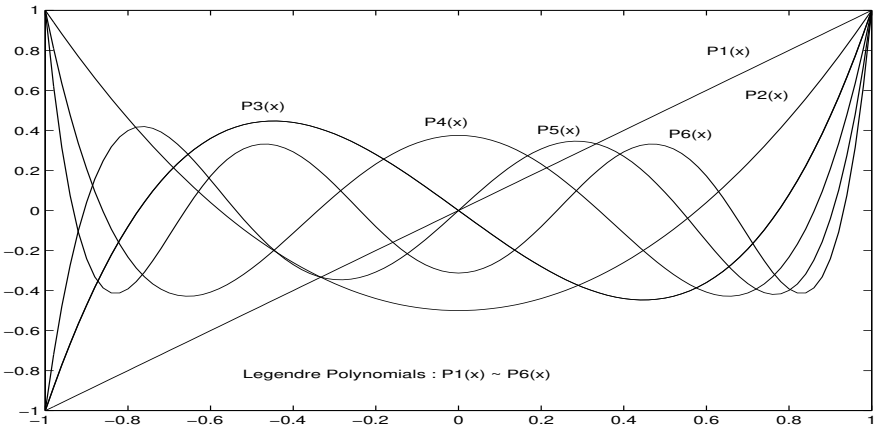
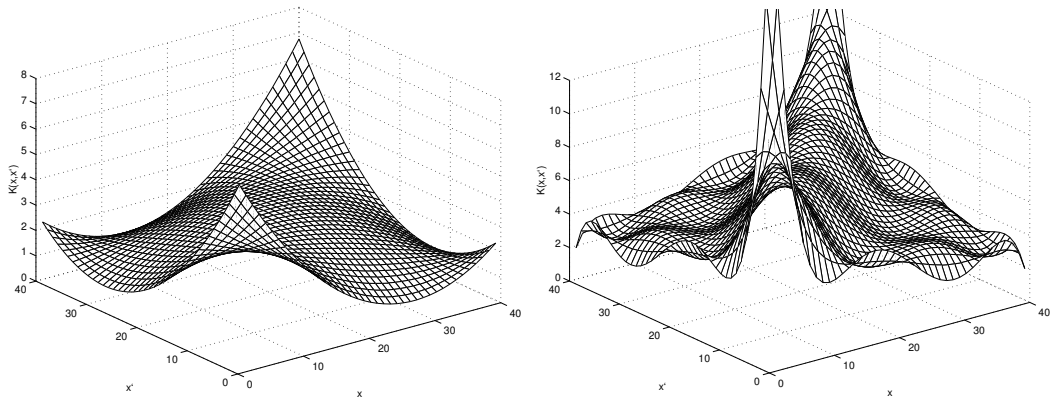


Figure 3.3: Legendre Polynomials : An example of orthogonal polynomials of increasing degree

Fig.3.3 plots the Legendre polynomials $P_1(t)$ till $P_6(t)$. Based on this set of OrthoNormal Bases (ONB), we can generate the reproducing kernel for a function subspace with the highest degree of polynomial N as

$$\begin{aligned} K(x,x') &= \sum_{n=0}^{N} e_n(x)e_n(x') \\ &= \begin{cases} \frac{(N+1)^2}{2(1-x^2)}[P_N(x)^2 - 2xP_N(x)P_{N+1}(x) + P_{N+1}(x)^2] & \text{if } x = x' \\ \frac{N+1}{2(x-x')}[P_{N+1}(x)P_N(x') - P_N(x)P_{N+1}(x')] & \text{otherwise.} \end{cases} \end{aligned} \tag{3.12}$$



(a) (b)

Figure 3.4: Reproducing kernels $K(x,x')$ for function spaces spanned by polynomial functions with (a) highest degree polynomial = 2 and (b) highest degree polynomial = 5

Fig.3.4 shows how the reproducing kernel looks graphically for spaces with highest polynomial degrees equals to two and five, respectively. The x and x' axis represent

©Sethu Vijayakumar 1998

the variables of the bi-variate reproducing kernel function and the z axis shows the value taken by these functions.

A family of such stratified function spaces can be used with novel model selection strategies as described in Schuurmans[Sch97] etc. An extensive analysis of Reproducing Kernel Hilbert Spaces (RKHS) including Sobolev spaces for different kinds of problem domains can be found in Saitoh[Sai97]. Alternate approaches of model selection under this framework, especially for real world problems like the learning of the sensorimotor map in a complex anthropomorphic robot arm by exploiting the knowledge of physical dynamical equations of motion, a strong apriori bias, will be addressed in Chapter 5.

©Sethu Vijayakumar 1998

4

Exact Incremental Learning for Optimal Generalization

4.1 Introduction

So far, the learning problem has been discussed under the basic framework of finding an optimum learning or reconstruction operator from a given set of training data [Oga90, ON84, OO86, YO92]. Under these circumstances, it was sufficient to construct a sampling operator A giving the sample value vector y and look at methods of finding an optimum learning operator X, as discussed in Chapter 2. However, when we consider the case when additional training data is made available, then, it is imperative to devise an efficient method to incorporate it into the learning scheme.

In this chapter, we will use Wiener learning as an example of the optimization criterion and formulate the concept of incremental learning. Then, using the properties of the generalized inverse of an non-negative definite matrix, we devise means of incrementally computing the learning operator and the learned function. We provide general incremental solutions for other optimization criteria and also, construct a concrete algorithm for incremental learning. Finally, some empirical tests in form of simulated learning in artificial examples are used to test the validity and effectiveness of the exact incremental learning technique devised here.

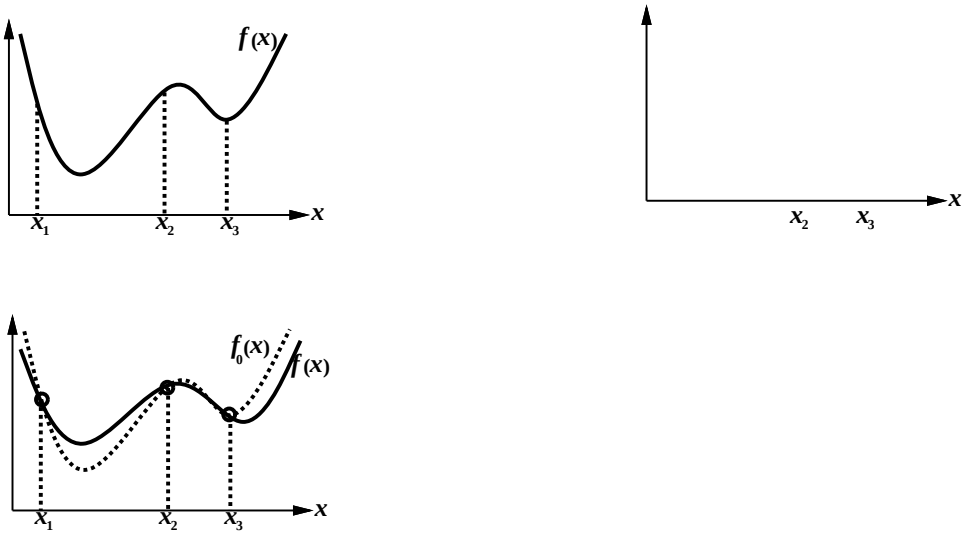


Figure 4.1: Neural network learning model. The two stages involved: (i) The selection of training data and hence, the sampling operator A, (ii) Evaluating the optimum learning operator X.

4.2 The concept of incremental learning

In a non-incremental or batch learning, sampling operator A is fixed based on the sampled training data using eqs.(2.4) and (2.7). With reference to Fig.4.1, it implies that the sampling operator A is already fixed based on the given training data and hence, the first stage of operations, in other words, the selection of training data is no longer a variable. The problem, therefore, has been discussed under the objective of finding an optimum learning operator in order to reconstruct the function f_0, which is as close to the original function f under some optimization criterion.

However, there are situations in which we can select the training data or when the training data comes in a continuous stream. Hence, another very important problem that has to be taken into account is the incorporation of the new training data. These kind of situations also arise when we are dealing with *active learning* in which the set of training data can be provided only one after the another and also uses the intermediate learned results for deciding on the next best sample.

One obvious method of incorporating the new data is by recalculating the learning operator all over again every time a new data comes in. This, we will refer to as the *non-incremental* method.

The problems involved in the non-incremental method is two-fold.

- Whenever a new training data is made available, this would involve recalculating the entire learning operator to incorporate this new information. This

©Sethu Vijayakumar 1998

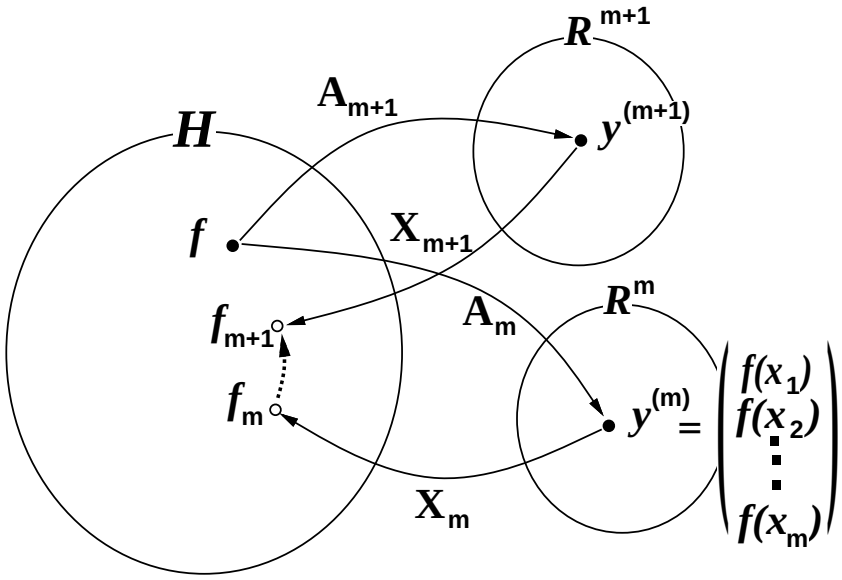


Figure 4.2: Incremental learning with additive training data

involves a lot of overhead.

- If the sampling itself is at a slow rate, then, we have to wait till all the training data samples have arrived before computing the learning operator.

With a view to solving these problems, we introduce a concept called the *incremental learning*. Incremental learning, here, refers to the successive or progressive learning with increase in the number of training data.

Since we will be dealing with a variable number of sample points and hence, a variable dimensional sampled space, we will use lower case letters to represent the dimension of the sampled space from now on. Let $\{e_n^{(m)} : 1 \leq n \leq m\}$ represent the natural basis of the sampled space R^m. Let A_m be the sampling operator from H to R^m while considering m sample points, defined as

$$A_m = \sum_{n=1}^{m} (e_n^{(m)} \otimes \overline{\psi_n}). \tag{4.1}$$

Let X_m be the corresponding learning operator for the m sample points derived using some optimization criterion. As shown in Fig.4.2, with the increasing number of training data corresponding to increasing indexes of the sampling operator A_m, the sampled space R^m keeps expanding and the reconstructed function f_m approaches the original function f. However, the point to be noted is that in the non-incremental method, the new learning operator X_{m+1} is computed directly from $\{x_n, y_n : 1 \leq n \leq m+1\}$ (Refer to Fig.4.2). This involves a lot of overhead and every time a

©Sethu Vijayakumar 1998

new data comes in, the entire learning operator has to be computed all over again. Suppose we are in the learning process, and every data point means a change in the best fit. We will fall hopelessly behind by trying every time to recalculate the new learning operator from scratch. What is needed is a method which looks at the changes in the learning operator produced by the new sampling operator and the new sampled value. Hence, in incremental approach, the new learning operator is computed from the previously derived learning operator X_m using m sample points, A_m and the next sampling location [1] x_{m+1}. This can be represented as

$$X_{m+1} = F(X_m, A_m, x_{m+1}), \tag{4.2}$$

where F is some function.

To show the necessity for such an approach, we present an example.

Assume a situation in which we are fitting a straight line through points that come from the radar. At any moment, we can stop and compute the best fit line. When the next point misses that line, the estimate has to be changed.

This kind of problem is analogous to the learning problem in the neural networks. We are trying to reconstruct the original function f from the given set of training data. As and when the new data arrives, the region of uncertainty grows smaller and the fit gets better and better. Summarizing the advantages of the incremental learning using functional analytic approach,

1. It is not necessary to recalculate the entire learning operator when a new training data is made available. Instead, results of the previous stage can be used for the computation, leading to a saving in the computational cost.

2. This approach can also be used in 'active learning' where training data is decided one after the other. Intermediate reconstruction results, as in f_m, are available. This information may be used to improve sampling efficiency by adaptive optimization criteria [Lut85, Mac92].

In the following Section, we shall talk about the Moore-Penrose generalized inverse and some special properties which would be useful in our incremental computation of the learning operator and function.

[1] The next sampling location is derived based on some heuristic or optimization criterion with a view to maximize the generalization ability for a minimized training set. This will be dealt with in detail in Chapter 6.

©Sethu Vijayakumar 1998

4.3 Moore-Penrose generalized inverse

In this section, we will define the concept of the Moore-Penrose generalized inverse and derive some general results which will be used as a tool in performing the incremental learning operation in the later part of this thesis.

A matrix has an inverse only if it is square, and even then only if it is non-singular, or in other words, if its columns and rows are linearly independent. However, even for singular and rectangular matrices, it is essential to compute an approximate inverse. To this effect, in 1955 Penrose showed that, for a finite matrix A (square or rectangular) of real or complex elements, there is a unique matrix X satisfying the four equations

$$AXA = A, \quad \text{'1-inverse'} \tag{4.3}$$

$$XAX = X, \quad \text{'2-inverse'} \tag{4.4}$$

$$(AX)^* = AX, \quad \text{'3-inverse'} \tag{4.5}$$

$$(XA)^* = XA \quad \text{'4-inverse'}, \tag{4.6}$$

where A^* denotes the conjugate transpose of A. This unique generalized inverse had previously been studied by E.H.Moore in 1920. Hence, it is commonly known as *Moore-Penrose inverse* and is often denoted by $A^\dagger$. Various useful properties of the generalized inverses are summarized below [Alb72]. Let I designate the $n \times n$ identity matrix. Then, we have, for arbitrary matrices,

(a) $A = 0(m \times n)$ implies $A^\dagger = 0(n \times m)$.

(b) $A^{\dagger\dagger} = A$.

(c) $A^{\dagger *} = A^{*\dagger}$.

(d) If A is non-singular,$A^\dagger = A^{-1}$.

(e) If A has full column rank, $A^\dagger = (A^*A)^{-1}A^*$.

(f) If A has full row rank, $A^\dagger = A^*(AA^*)^{-1}$.

(g) If U and V are unitary, $(UAV)^\dagger = V^*A^\dagger U^*$.

(h) If A is a projection matrix, $A^\dagger = A$.

(i) $A^\dagger A, AA^\dagger, I - A^\dagger A$ and $I - AA^\dagger$ are all projection matrices.

(j) $EF = 0$ implies $F^\dagger E^\dagger = 0$.

(k) If P is a projection matrix, $(PQ)^\dagger = Q^\dagger P$ whenever either $PQ = Q$ or P commutes with $Q, Q^\dagger Q$ and $QQ^\dagger$.

©Sethu Vijayakumar 1998

(1) The equation $XA = B$ has a solution if and only if

$$BA^{\dagger}A = B, \tag{4.7}$$

in which case X is a solution if and only if there is a matrix Y such that

$$X = BA^{\dagger} + Y(I - AA^{\dagger}). \tag{4.8}$$

The solution is unique if and only if the null space of A is void, i.e., $(I - AA^{\dagger} = 0)$.

The above theory of generalized inverses, defined here for matrices, is applicable in exactly the same form for operators with a closed domain. We shall now present a few results on the Moore-Penrose generalized inverse of a special kind of matrix. The matrix must be non-negative definite and should have a particular structure.

Non-negative definiteness is explained using the following lemma [Alb72].

Lemma 4.1 (Non-negative definiteness) *For any symmetric matrix M, the following are equivalent conditions for non-negative definiteness.*

1. $(Mf, f) \geq 0$ *for any function f.*
2. $M = HH^T$ *for some matrix H.*
3. $(x, Mx) \geq$ *for all vectors x.*
4. *The eigen values of M are all non-negative.*
5. *There is a symmetric matrix R, such that $R^2 = M$.*

Non-negative definiteness of M is expressed by $M \geq 0$.

Let V_m be an $m \times m$ *non-negative definite matrix*. Let V_{m+1} be represented in the form

$$V_{m+1} = \begin{pmatrix} V_m & s_m \\ s_m^T & \sigma \end{pmatrix}, \tag{4.9}$$

where s_m and σ represent an m-dimensional vector and scalar, respectively. We assume that $V_0 = 0$, $s_0 = 0$. Let t_m and α be defined as

$$t_m = V_m^{\dagger} s_m, \tag{4.10}$$

$$\alpha = \sigma - (t_m, s_m). \tag{4.11}$$

Lemma 4.2 (Generalized inverse of partitioned matrix) *[Alb72] If V_{m+1} is non-negative definite, then, $\alpha \geq 0$ and the Moore-Penrose generalized inverse of V_{m+1} for $\alpha > 0$ is given as*

$$V_{m+1}^{\dagger} = \begin{bmatrix} V_m^{\dagger} + \frac{1}{\alpha}(t_m \otimes \overline{t_m}) & -\frac{1}{\alpha} t_m \\ -\frac{1}{\alpha} t_m^T & \frac{1}{\alpha} \end{bmatrix}. \tag{4.12}$$

©Sethu Vijayakumar 1998

The result for the condition $\alpha = 0$ has been omitted because this condition works out to be a trivial case in context to the learning problem, as will be proved later.

4.4 Geometric interpretation of Wiener learning

As an example of the optimization criterion for incremental learning, we shall use the Wiener criterion for the major part of the analysis. Results on other optimization criterion, which can be derived on the same lines, will be given in Section 4.8.

In this section, we will show that the optimization or learning with the Wiener criterion has a geometric interpretation in terms of oblique projections onto a subspace, a result we will use in our analysis later. To begin with, we give a few lemmas which lead to that result here.

Lemma 4.3 (Alternate condition for Wiener criterion) *[YO92] A necessary and sufficient condition for an operator X to satisfy Wiener criterion is*

$$XAf = f : f \in \mathcal{R}(RA^*). \tag{4.13}$$

Proof of this lemma is given in Appendix A4.1.

So, we can now use this alternate condition for future proofs. It is known [YO92] that almost all learnable functions in case of Wiener learning belongs to the range $\mathcal{R}(R)$. The image of a subspace L due to an operator A is denoted as

$$AL = \{Af | f \in L\}. \tag{4.14}$$

Then, we have a set of following results on the space $\mathcal{R}(R)$.

Lemma 4.4 *[YO92] For any closed subspaces L and any non-negative operator R, it holds that*

$$L + RL^{\perp} \supset \mathcal{R}(R). \tag{4.15}$$

Since $\mathcal{N}(A)$ is a closed subspace, substituting $L = \mathcal{N}(A)$ into Lemma 4.4, we have

$$\mathcal{N}(A) + R\mathcal{N}(A)^{\perp} \supset \mathcal{R}(R). \tag{4.16}$$

Since $R\mathcal{N}(A)^{\perp} = \mathcal{R}(RA^*)$, it follows from eq.(4.16) that

$$\mathcal{N}(A) + \mathcal{R}(RA^*) \supset \mathcal{R}(R). \tag{4.17}$$

Then we have

©Sethu Vijayakumar 1998

Lemma 4.5 (Decomposition of the space $\mathcal{R}(R)$) *[YO92] It holds that*

$$\mathcal{R}(R) = \mathcal{R}(RA^*) + (\mathcal{R}(R) \cap \mathcal{N}(A)). \tag{4.18}$$

Lemma 4.5 shows that it is possible to express the original function space for Wiener learning $\mathcal{R}(R)$ as the sum of the subspaces $\mathcal{R}(RA^*)$ of Lemma 4.3 and $\mathcal{R}(R) \cap \mathcal{N}(A)$. Also, all the functions belonging to $\mathcal{R}(R) \cap \mathcal{N}(A)$ yields zero on being sampled by the sampling operator A. Hence, the reconstructed function also becomes zero for the original function belonging to $\mathcal{R}(R) \cap \mathcal{N}(A)$. So we have the following result.

Lemma 4.6 (Alternate condition for Wiener Learning: II) *The necessary and sufficient condition for an operator X to satisfy Wiener criterion within subspace $\mathcal{R}(R)$ is that XA is a projection on $\mathcal{R}(RA^*)$ along $\mathcal{R}(R) \cap \mathcal{N}(A)$.*

By using the result of eq.(4.17), we can extend the subspace of the original functions in Lemma 4.6 to $\mathcal{R}(RA^*) + \mathcal{N}(A)$. Hence we have

Lemma 4.7 (Geometric Interpretation of Wiener Learning) *For a learning operator X satisfying the Wiener criterion, XA provides an orthogonal projection onto $\mathcal{R}(RA^*)$ along $\mathcal{N}(A)$ within the subspace $(\mathcal{R}(RA^*) + \mathcal{N}(A))$.*

In other words, the Wiener learning can be called as an oblique projection onto the subspace $\mathcal{R}(RA^*)$ along $\mathcal{R}(R) \cap \mathcal{N}(A)$ within the subspace $(\mathcal{R}(RA^*) + \mathcal{N}(A))$ which is a superset of $\mathcal{R}(R)$. However, we assume that the possible set of original functions defined by the Hilbert space H is already contained in the $\mathcal{R}(R)$ because of the correlation operator. Hence, Lemma 4.7 can be interpreted as the following final result on Wiener learning.

Theorem 4.1 (Geometric Interpretation of Wiener Learning) *A learning operator X satisfying the Wiener criterion provides an oblique projection XA of the original function onto a subspace $\mathcal{R}(RA^*)$ along $\mathcal{R}(R) \cap \mathcal{N}(A)$.*

This result is graphically depicted in Fig.4.3. This interpretation of the Wiener learning will be used later in the chapter while analyzing the conditions that arise in the incremental computation of the learning operator and function. The projection criterion, by definition, results in the orthogonal projection onto a subspace $\mathcal{R}(A^*)$. The relationship between the two optimization criterion in the noiseless case can, hence, be shown graphically as in Fig.4.4. In the next section, we will formalize the Wiener optimization criterion in the incremental context.

©Sethu Vijayakumar 1998

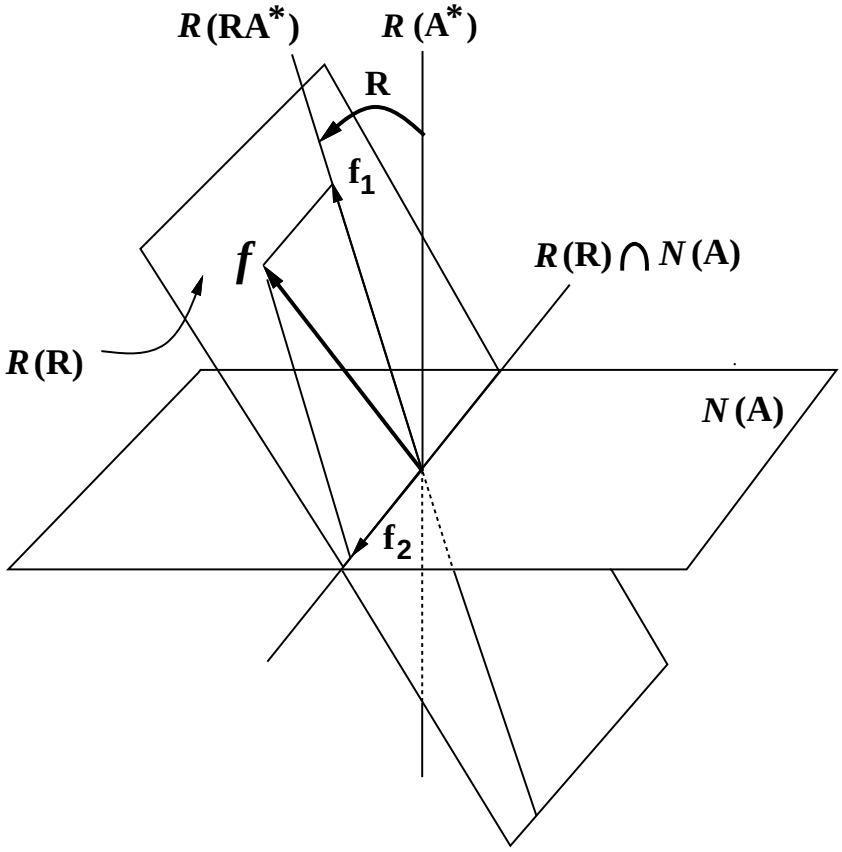


Figure 4.3: Geometrical interpretation of a Wiener learning. For an operator X satisfying the Wiener criterion, XA provides an oblique projection f_1 of $f \in \mathcal{R}(R)$ onto $\mathcal{R}(RA^*)$ along $\mathcal{R}(R) \cap \mathcal{N}(A)$.

4.5 The incremental Wiener criterion

The Wiener criterion, described in Section 2.3.3, can be represented in the context of incremental learning as follows.

Definition 4.1 (Incremental Wiener Learning Criterion) *Minimize the following functional with respect to X_{m+1}:*

$$J_W[X_{m+1}] = E_f \|X_{m+1} A_{m+1} f - f\|^2, \tag{4.19}$$

where $\|.\|$ is the norm in the original function space H and E_f is the average of f over the original function space.

Considering a particular solution for the Wiener learning, i.e., taking $W = 0$ in eq.(2.21), we have

$$X_{m+1} = RA^*_{m+1} U^{\dagger}_{m+1}, \tag{4.20}$$

©Sethu Vijayakumar 1998

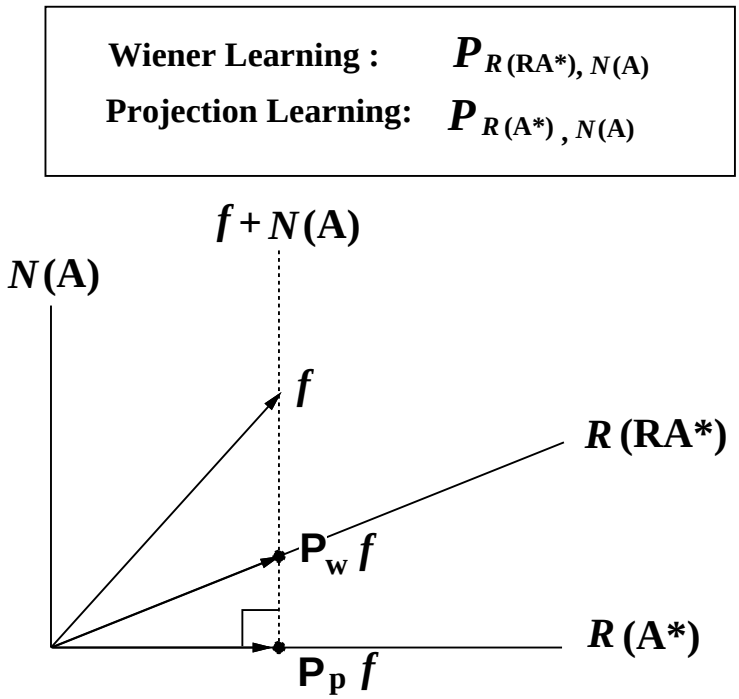


Figure 4.4: Relationship between the Wiener and Projection learning operators. Wiener learning is an oblique projection onto $\mathcal{R}(RA^*)$ whereas Projection learning is an orthogonal projection onto $\mathcal{R}(A^*)$, both along $\mathcal{N}(A)$.

where

$$U_{m+1} = A_{m+1} R A^*_{m+1}. \tag{4.21}$$

In the next section, we shall apply Lemma 4.2 to our incremental computation and derive closed form solutions for the learning operator and the learned function.

4.6 Incremental computation of the learning operator

In this section, we shall formulate a method of incorporating the newly available training data to the already computed learning operator using the previously computed results and the newly added training point.

In order to express the new sampling operator A_{m+1} in terms of the old sampling operator A_m, we introduce an operator Γ defined as

$$\Gamma = \sum_{n=1}^{m} (e_n^{(m+1)} \otimes \overline{e_n^{(m)}}). \tag{4.22}$$

©Sethu Vijayakumar 1998

It is easy to see that Γ is an operator transforming the bases of R^m into the bases of R^{m+1}:

$$\Gamma e_p^{(m)} = e_p^{(m+1)} : 1 \leq p \leq m. \tag{4.23}$$

The following is a direct result of eq.(4.23).

Lemma 4.8 *Let u be an m-dimensional vector, i.e. $u \in R^m$ with elements $u = (u_1 u_2 ... u_m)^T$. Then, $\Gamma u \in R^{m+1}$ is always of the form $\Gamma u = (u_1 u_2 ... u_m 0)^T$.*

Hence, the operator Γ is an operator transforming the sample vector y from R^m to R^{m+1} (refer Fig.4.5)

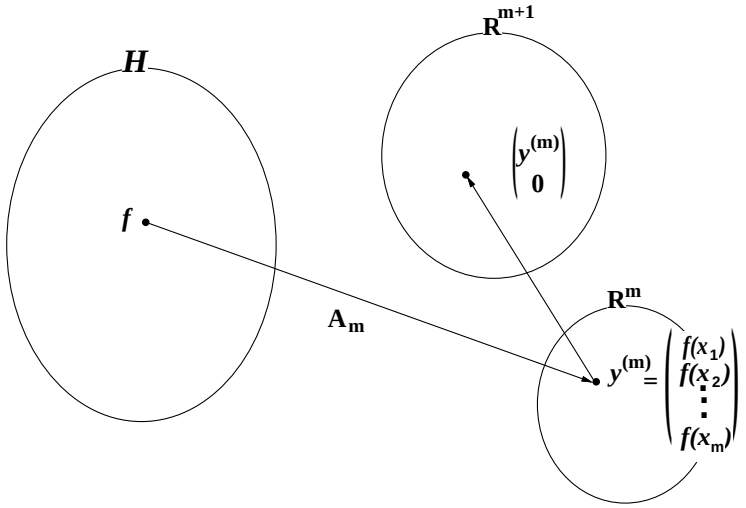


Figure 4.5: Operator Γ mapping $y^{(m)}$ from R^m into R^{m+1} space

Lemma 4.9 *Let V be an $m \times m$ matrix with elements $V = (v_{pq} : 1 \leq p, q \leq m)$. Then, $\Gamma V \Gamma^*$ represents an $(m+1) \times (m+1)$ matrix V' with elements (v'_{pq}) defined as*

$$\begin{aligned} v'_{pq} &= v_{pq} : 1 \leq p, q \leq m, \\ &= 0 \ : \ Otherwise. \end{aligned}$$

The proof of this lemma is provided in Appendix A4.2.

Now, we can write the new sampling operator A_{m+1} as

$$A_{m+1} = \Gamma A_m + (e_{m+1}^{(m+1)} \otimes \overline{\psi_{m+1}}) \tag{4.24}$$

because of eq.(4.1) and Lemma 4.8.

©Sethu Vijayakumar 1998

Let

$$s_m = A_m R\psi_{m+1}, \tag{4.25}$$
$$\sigma = (R\psi_{m+1})(x_{m+1}), \tag{4.26}$$
$$t_m = U_m^{\dagger} s_m, \tag{4.27}$$
$$\alpha = \sigma - (t_m, s_m). \tag{4.28}$$

It follows from eqs.(4.21) and (4.24) that

$$U_{m+1} = \Gamma U_m \Gamma^* + (\Gamma s_m \otimes \overline{e_{m+1}^{(m+1)}}) + (e_{m+1}^{(m+1)} \otimes \overline{\Gamma s_m}) + \sigma(e_{m+1}^{(m+1)} \otimes \overline{e_{m+1}^{(m+1)}}). \tag{4.29}$$

Looking at different terms of the expansion, it is easy to show through Lemmas 4.8 and 4.9 that the matrix U_{m+1} can be written in the form

$$U_{m+1} = \begin{pmatrix} U_m & s_m \\ s_m^T & \sigma \end{pmatrix}, \tag{4.30}$$

as shown in Fig.4.6, which also corresponds to the structure of eq.(4.9). It can also be proved that the matrix represented by the terms U_{m+1} is *non-negative definite* because of eq.(4.21). Therefore, Lemma 4.2 can be applied to calculate the term $U_{m+1}^{\dagger}$ recursively for the condition $\alpha > 0$. However, we still have to consider the case when $\alpha = 0$. We will now present a few lemmas to be used in the analysis.

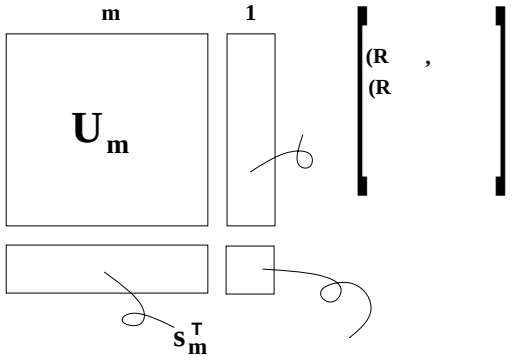


Figure 4.6: Structure of the matrix U_{m+1}

Let S_m and S_{mc} be defined as

$$S_m = \mathcal{R}(RA_m^*), \tag{4.31}$$
$$S_{mc} = \mathcal{R}(R) \cap \mathcal{N}(A_m), \tag{4.32}$$

Then, $\mathcal{R}(R)$ can be decomposed using the direct sum decomposition based on Lemma 4.5 as

$$\mathcal{R}(R) = \mathcal{R}(RA^*) \dot{+} \mathcal{R}(R) \cap \mathcal{N}(A)$$
$$= S_m \dot{+} S_{mc}. \tag{4.33}$$

©Sethu Vijayakumar 1998

It should be noted here that the direct sum decomposition operator, denoted by $\dot{+}$, represents a non-orthogonal decomposition in general. Also, it is known from Theorem 4.1 that in the case of Wiener learning, the learned function $f_m = X_m A_m f$, for every $f \in \mathcal{R}(R)$, is an oblique projection[2] of f onto S_m along S_{mc}, that is,

$$f_m = X_m A_m f = P_{S_m, S_{mc}} f. \tag{4.34}$$

In general, $S_{m+1} \supseteq S_m$. However,using eqs.(4.33) and(4.34), it is easy to prove the following lemma.

Lemma 4.10 *[Vij95] If $S_{m+1} = S_m$, then the newly learned function does not differ from the previously learned function, i.e., $f_{m+1} = f_m$. On the other hand, if $S_{m+1} \supset S_m$, then $f_{m+1} \neq f_m$, in general.*

Lemma 4.11 *It holds that*

$$rank(U_{m+1}) = rank(U_m) + sgn(\alpha). \tag{4.35}$$

The proof of the above lemma is provided in Appendix A4.3.

Using Lemma 4.11, the non-negative definiteness of R and the definition of U_{m+1} in eq.(4.21), we can derive the following lemma.

Lemma 4.12 *It holds that*

$$dim(S_{m+1}) = dim(S_m) + sgn(\alpha). \tag{4.36}$$

Since Lemma 4.12 deals with finite dimensional subspaces, the equality of the dimensions of S_{m+1} and S_m means the oneness of the subspaces. Hence, we have the following result.

Lemma 4.13 *$S_{m+1} = S_m$ if and only if $\alpha = 0$. Also, $S_{m+1} \supset S_m$ if and only if $\alpha > 0$.*

Lemma 4.13 and Lemma 4.10 imply that for the case when $\alpha = 0$, we need not carry out learning for such data since the learned function does not vary even after training with the new data, i.e., $f_{m+1} = f_m$. Therefore, we will now deal with the case $\alpha > 0$ only.

[2] An oblique projection is represented as $P_{B,C}$, which refers to the projection onto a subspace B along a subspace C.

©Sethu Vijayakumar 1998

Theorem 4.2 *Let s_m, σ, t_m and α be defined as in eqs.(4.25), (4.26), (4.27) and (4.28),respectively. If $\alpha > 0$, then the new learning operator X_{m+1} can be computed incrementally from the previous learning operator X_m as*

$$X_{m+1} = (X_m - Y_m)\Gamma^* + (v_m \otimes \overline{e_{m+1}^{(m+1)}}), \tag{4.37}$$

where

$$v_m = \frac{1}{\alpha}(R\psi_{m+1} - RA_m^* t_m), \tag{4.38}$$

$$Y_m = (v_m \otimes \overline{t_m}), \tag{4.39}$$

Proof: It follows from eqs.(4.20) and (4.24) that

$$X_{m+1} = R[\Gamma A_m + (e_{m+1}^{(m+1)} \otimes \overline{\psi_{m+1}})]^* U_{m+1}^\dagger. \tag{4.40}$$

Since U_{m+1} in eq.(4.40) is non-negative definite and has the structure as shown in Fig.5, we can apply Lemma 4.2 to eq.(4.40). Hence, we have

$$X_{m+1} = R[A_m^*\Gamma^* + (\psi_{m+1} \otimes \overline{e_{m+1}^{(m+1)}})] \begin{bmatrix} U_m^\dagger + \Xi & \rho \\ \rho^T & \tau \end{bmatrix}, \tag{4.41}$$

where Ξ, ρ, and τ are defined as below.

$$\Xi = \frac{1}{\alpha}(t_m \otimes \overline{t_m}), \tag{4.42}$$

$$\rho = -\frac{1}{\alpha} t_m, \tag{4.43}$$

$$\tau = \frac{1}{\alpha}. \tag{4.44}$$

Applying Lemmas 4.8 and 4.9 to eq.(4.41), we have

$$\begin{aligned} X_{m+1} = {} & R[A_m^*\Gamma^* + (\psi_{m+1} \otimes \overline{e_{m+1}^{(m+1)}})][\Gamma(U_m^\dagger + \Xi)\Gamma^* + (\Gamma\rho \otimes \overline{e_{m+1}^{(m+1)}}) \\ & + (e_{m+1}^{(m+1)} \otimes \overline{\Gamma\rho}) + \tau(e_{m+1}^{(m+1)} \otimes \overline{e_{m+1}^{(m+1)}})]. \end{aligned} \tag{4.45}$$

Since $\Gamma^*\Gamma = I$ and $\Gamma^* e_{m+1}^{(m+1)} = 0$, it follows from eqs.(4.45)and (4.20) that

$$\begin{aligned} X_{m+1} = {} & RA_m^*(U_m^\dagger + \Xi)\Gamma^* + (RA_m^*\rho \otimes \overline{e_{m+1}^{(m+1)}}) + (R\psi_{m+1} \otimes \overline{\Gamma\rho}) \\ & + \tau(R\psi_{m+1} \otimes \overline{e_{m+1}^{(m+1)}}) \\ = {} & (X_m + RA_m^*\Xi + (R\psi_{m+1} \otimes \overline{\rho}))\Gamma^* + (RA_m^*\rho \\ & + \tau R\psi_{m+1}) \otimes \overline{e_{m+1}^{(m+1)}}. \end{aligned} \tag{4.46}$$

©Sethu Vijayakumar 1998

Substituting eqs.(4.42),(4.43), and (4.44) into eq.(4.46), we have

$$\begin{aligned} X_{m+1} &= X_m\Gamma^* + [\frac{1}{\alpha}(RA_m^*(t_m \otimes \overline{t_m}) - R(\psi_{m+1} \otimes \overline{t_m}))]\Gamma^* u \\ &\quad + [\frac{-1}{\alpha}(RA_m^* t_m - R\psi_{m+1}) \otimes \overline{e_{m+1}^{(m+1)}}] \\ &= [X_m - (v_m \otimes \overline{t_m})]\Gamma^* + (v_m \otimes \overline{e_{m+1}^{(m+1)}}) \\ &= (X_m - Y_m)\Gamma^* + (v_m \otimes \overline{e_{m+1}^{(m+1)}}). \end{aligned} \tag{4.47}$$

Hence, the theorem holds.

4.7 Incremental computation of the learned function

Theorem 4.3 *When $\alpha > 0$, the learned function f_{m+1} due to $m+1$ sample points can be computed incrementally from the previous learned function f_m and the new sample value $f(x_{m+1})$ as*

$$f_{m+1} = f_m - Y_m y^{(m)} + f(x_{m+1})v_m, \tag{4.48}$$

where Y_m and v_m are given by eqs.(4.39) and (4.38), respectively.

Proof: From the general form of eq.(2.9), we can write the newly learned function f_{m+1} as

$$f_{m+1} = X_{m+1} y^{(m+1)}. \tag{4.49}$$

Using eq.(4.37) of Theorem 4.2 to compute X_{m+1} in eq.(4.49), we have

$$\begin{aligned} f_{m+1} &= (X_m - Y_m)\Gamma^* y^{(m+1)} + (v_m \otimes \overline{e_{m+1}^{(m+1)}}) y^{(m+1)} \\ &= (X_m - Y_m) y^{(m)} + (y^{(m+1)}, e_{m+1}^{(m+1)}) v_m, \end{aligned}$$

which yields eq.(4.48).

On reducing eq.(4.48), we can further clarify the relationship between f_m and f_{m+1}, which we will proceed to do in the next theorem. As shown in Theorem 4.1, f_m is a projection of f onto S_m along S_{mc}. Let f_{mc} be a projection of f onto S_{mc} along S_m.

Now, consider a function ϕ defined as

$$\phi \equiv R\psi_{m+1}. \tag{4.50}$$

Let ϕ_m and ϕ_{mc} be the projection of ϕ onto S_m and S_{mc} along S_{mc} and S_m, respectively. Then, it can be proved, as shown in Appendix A4.4, that α of eq.(4.11) is equivalent to the value of the function ϕ_{mc} at the new sampled location x_{m+1}, that is,

©Sethu Vijayakumar 1998

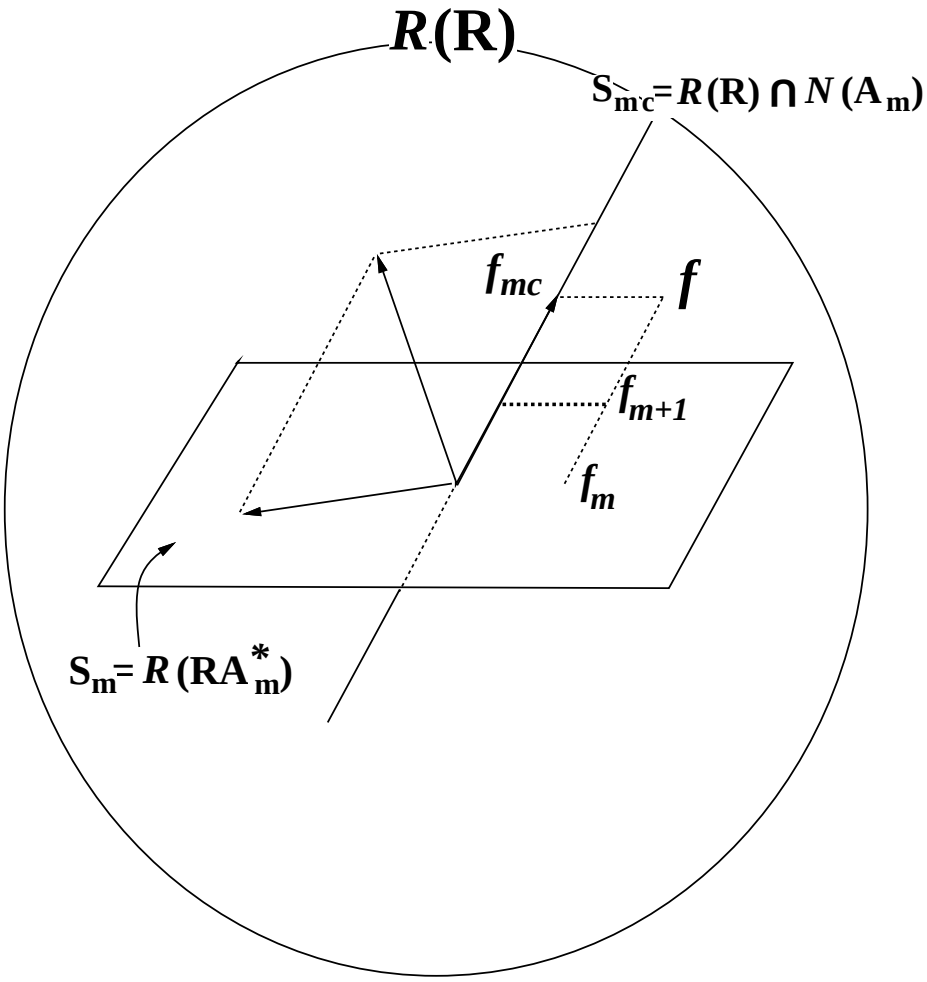


Figure 4.7: Relationship between f_m and f_{m+1}.

Lemma 4.14

$$\alpha = \phi_{mc}(x_{m+1}). \tag{4.51}$$

Using these relations and further reducing eq.(4.48), we have the following theorem.

Theorem 4.4 *When $\alpha > 0$, it holds that*

$$f_{m+1} = f_m + \frac{f_{mc}(x_{m+1})}{\phi_{mc}(x_{m+1})}\phi_{mc}. \tag{4.52}$$

A proof of the theorem is provided in Appendix A4.5.

A schematic diagram of the relationship is shown in Fig.4.7. It is clear from eq.(4.52) and Fig.4.7 that the newly learned function f_{m+1} can be expressed as the sum of f_m and a scalar multiple of the function ϕ_{mc}. Here, ϕ_{mc} is the projection of ϕ onto S_{mc},

©Sethu Vijayakumar 1998

which again depends on the location of the new sample due to eqs.(4.50) and (2.4). Another point to be noted is that the term $f_{mc}(x_{m+1})$ which appears in the scalar part of eq.(4.52) is, in fact, the error between the actual function and the learned function at the new sample point, i.e.,

$$f_{mc}(x_{m+1}) = (f - f_m)(x_{m+1}). \tag{4.53}$$

An explicit relationship between the newly sampled location x_{m+1}, the newly learned function and the previous function provides a basis for the work on active data selection for improving generalization.

The next section provides a general result for incremental computations under different optimization criteria.

4.8 General incremental learning results

Using a similar approach as adopted here, we can find incremental closed form solutions for the general class of optimization criterion. First we provide incremental learning results for the projection and memorization learning criterion.

Theorem 4.5 (Incremental learning operator : projection learning) *[Nak97] The incremental learning operator for projection learning can be computed as*

$$X_{m+1}^{(P)} = (X_m^{(P)} - v_m^{(P)} \otimes \overline{t_m^{(P)}})\Gamma^* + v_m^{(P)} \otimes \overline{e_{m+1}^{(m+1)}}, \tag{4.54}$$

where $v_m^{(P)}$, $t_m^{(P)}$ and $s_m^{(P)}$ are defined as

$$v_m^{(P)} = \frac{1}{\alpha}(\psi_{m+1} - A_m^* t_m^{(P)}), \tag{4.55}$$

$$t_m^{(P)} = (A_m A_m^*)^\dagger s_m^{(P)}, \tag{4.56}$$

$$s_m^{(P)} = A_m \psi_{m+1}. \tag{4.57}$$

Theorem 4.6 (Incrementally learned function : projection learning) *[Nak97] The function learned by projection learning using $m+1$ training data, f_{m+1}, can be incrementally computed as*

$$f_{m+1} = f_m + \frac{y_{m+1} - f_m(x_{m+1})}{\psi_{mc}(x_{m+1})}\psi_{mc}, \tag{4.58}$$

where ψ_{mc} is the orthogonal projection of ψ_m onto $\mathcal{N}(A_m)$.

Theorem 4.7 (Incremental learning operator : memorization learning) *[Nak97] The incremental learning operator for memorization learning can be computed as*

$$X_{m+1}^{(M)} = (X_m^{(M)} - \zeta_m \otimes \overline{h^{(m)}})\Gamma^* + \gamma_m \otimes \overline{e_{m+1}^{(m+1)}}, \tag{4.59}$$

©Sethu Vijayakumar 1998

where ζ_m, γ_m and $h^{(m)}$ are defined as

$$\zeta_m = \frac{1}{\alpha}(\psi_m - A_m^* t_m^{(M)}), \tag{4.60}$$

$$\gamma_m = \tilde{\eta}_m + (1 - \tilde{\eta}_m(x_m + 1))\zeta_m, \tag{4.61}$$

$$h^{(m)} = t_m^{(M)} - K_m^\dagger K_m(\Omega_m^* t_m^{(M)} - q_m). \tag{4.62}$$

Refer to [Nak97] for the undefined terms here.

Theorem 4.8 (Incrementally learned function : memorization learning) *[Nak97] The function learned by projection learning using $m+1$ training data, f_{m+1}, can be incrementally computed as*

$$f_{m+1} = f_m + y_{m+1}\eta_{mc} + \frac{\beta - y_{m+1}\eta_{mc}}{\alpha}\psi_{mc} \tag{4.63}$$

where ψ_{mc} is the orthogonal projection of ψ_m onto $\mathcal{N}(A_m)$. For rest of the notations, refer [Nak97].

The general form of the incremental learning operator X_{m+1}, summarised in Table 4.1, is given using functions ζ_m, γ_m and the vector $h^{(m)}$ as

$$X_{m+1} = (X_m - \zeta_m \otimes \overline{h^{(m)}})\Gamma^* + \gamma_m \otimes \overline{e_{m+1}^{(m+1)}}, \tag{4.64}$$

where ζ_m, γ_m and $h^{(m)}$ are defined for the respective cases. The incremental computation of learned functions for different optimization criterion are also summarised in Table.4.2.

4.9 Pseudo code of incremental learning algorithm

In this section, we will give the pseudocode for implementation of the algorithm introduced in Section 4.7, which corresponds to Wiener learning. For other optimization criterion, the corresponding equations can be substituted from the Tables.4.1 and 4.2. The pseudocode for implementation of incremental learning for optimal generalization is as follows.

1. `Decide on optimal search space` H `based on model selection techniques described in Chapter 3.`

2. `Construct reproducing kernel.`

   (a) `Use existing reproducing kernel for` H`, if known.`
   `For example, for band-limited function spaces, use eq.(3.7).`

©Sethu Vijayakumar 1998

Table 4.1: Incremental computation of learning operators for various optimization criterion

	$X_{m+1} = (X_m - \zeta_m \otimes \overline{h^{(m)}})\Gamma^* + \gamma_m \otimes \overline{e_{m+1}^{(m+1)}}$
Wiener Learning	$\zeta_m = \frac{1}{\alpha}(\phi - RA_m^* t_m)$ $\gamma_m = \zeta_m$ $h^{(m)} = t_m$; refer eq.(4.27)
Projection Learning	$\zeta_m = \frac{1}{\alpha}(\psi_{m+1} - A_m^* t_m^{(P)})$; refer eq.(4.56) $\gamma_m = \zeta_m$ $h^{(m)} = (A_m A_m^*)^\dagger s_m^{(P)}$; refer eq.(4.57)
Memorization Learning	$\zeta_m = \frac{1}{\alpha}(\psi_{m+1} - A_m^* t_m^{(M)})$ $\gamma_m = \tilde{\eta_m} + (1 - \tilde{\eta}_m(x_{m+1}))\zeta_m$ $h^{(m)} = t_m^{(M)} - K_m^\dagger K_m(\Omega_m^* t_m^{(M)} - q_m)$

Table 4.2: Incrementally learned functions under different optimization criteria

Wiener Learning	$f_{m+1} = f_m + \frac{y_{m+1} - f_m(x_{m+1})}{\phi_{mc}(x_{m+1})}\phi_{mc}.$
Projection Learning	$f_{m+1} = f_m + \frac{y_{m+1} - f_m(x_{m+1})}{\psi_{mc}(x_{m+1})}\psi_{mc}.$
Memorization Learning	$f_{m+1} = f_m + y_{m+1}\tilde{\eta_m} + \frac{\beta - y_{m+1}\tilde{\eta_m}}{\alpha}\psi_{mc}.$

©Sethu Vijayakumar 1998

(b) Else, construct an OrthoNormal Basis(ONB) $\{\varphi_n\}_{n=1}^{N}$ for the Hilbert space H. Then, compute the reproducing kernel corresponding to the Hilbert space H as

$$K(x,x') = \sum_{n=1}^{N} \varphi_n(x)\varphi_n(x').$$

3. If using the Wiener criterion, approximate the correlation operator R.

4. Initialize to zero function : $f_0 = 0$.

5. For each new training data (x_{m+1}, y_{m+1}),

   (i) Use the reproducing kernel $K(x,x')$ to instantiate $\psi_{m+1} = K(x, x_{m+1})$.

   (ii) Compute $\phi = R\psi_{m+1}$.

   (iii) Compute oblique projection $\phi_{mc} = P_{\mathcal{N}(A_m),\mathcal{R}(RA_m^*)}\phi$.

   (iv) Compute $f_{mc}(x_{m+1})$ as $f_{mc}(x_{m+1}) = f(x_{m+1}) - f_m(x_{m+1}) = y_{m+1} - f_m(x_{m+1})$.

6. Compute the newly learned function f_{m+1} as

$$f_{m+1} = f_m + \frac{f_{mc}(x_{m+1})}{\phi_{mc}(x_{m+1})}\phi_{mc}.$$

7. If no more training data to incorporate, STOP.
   Else, goto step 5.

4.10 Simulation results and experiments

In this section, we will perform computer simulations with specific objectives. In all the experiments, we will use the incremental learning rule derived in the previous section to compute the successively learned functions. The first set of experiments will merely demonstrate the fact that generalizing ability (measured in terms of the L_2-norm error between the original and learned function) of the functions get better and better as we add more training examples. In the second set of experiments, we show that the selection of appropriate training data is essential to ensure better generalization. Finally, the differences arising out of model selection dilemma is depicted through an example in the third set of experiments.

©Sethu Vijayakumar 1998

4.10.1 The generalization capability

Let us consider a Hilbert space spanned by the linear combinations of the functions

$$\{1, sin(x), sin(2x), sin(3x), sin(4x), sin(5x)\}. \tag{4.65}$$

The inner product in this space is defined as

$$(f, g) = \frac{1}{2\pi} \int_0^{2\pi} f(x)g(x)dx. \tag{4.66}$$

Since each functions in eq.(4.65) is orthogonal to each other and

$$\|1\| = 1, \quad \|sin(nx)\| = \frac{1}{\sqrt{2}} \text{ for all } n, \tag{4.67}$$

the orthonormal bases(ONB) of the space H, $\{\varphi_n\}_{n=1}^6$, is given as

$$\{\varphi_n\}_{n=1}^6 = \{1, \sqrt{2}sin(x), \sqrt{2}sin(2x), \sqrt{2}sin(3x), \sqrt{2}sin(4x), \sqrt{2}sin(5x)\}. \tag{4.68}$$

Hence, based on eq.(3.3), the reproducing kernel of H can be written as

$$K(x, x') = 1 + 2\sum_{n=1}^{5} sin(nx')sin(nx). \tag{4.69}$$

Let us consider a problem of learning a function f belonging to H and given as

$$f(x) = 4 - sin(x) + sin(2x) - sin(3x) + sin(4x) - sin(5x). \tag{4.70}$$

This function is represented with a dotted line in the Fig.4.8. We consider learning with training data $\{x_m, y_m\}_{m=1}^6$ using the Projection optimization criterion.

{Data #1}
Assume the first training data is given as $(x_1, y_1) = (\frac{\pi}{4}, 5 - \frac{1}{\sqrt{2}})$. Then, the sampling function $\psi_1(x)$, based on eqs.(2.4) and (4.69) is given as

$$\psi_1(x) = 1 + \sqrt{2}sin(x) + 2sin(2x) + \sqrt{2}sin(3x) - \sqrt{2}sin(5x). \tag{4.71}$$

Following the steps of the pseudocode, we can compute the learned function due to the first training data as

$$f_1(x) = \frac{10 - \sqrt{2}}{12}(1 + \sqrt{2}sin(x) + 2sin(2x) + \sqrt{2}sin(3x) - \sqrt{2}sin(5x)). \tag{4.72}$$

The resultant function is plotted in Fig.4.8(a) by a solid line. The generalization error, computed as the Euclidean normed distance between the original and learned function works out to be

$$\|f_1 - f\| = 3.9279 \tag{4.73}$$

©Sethu Vijayakumar 1998

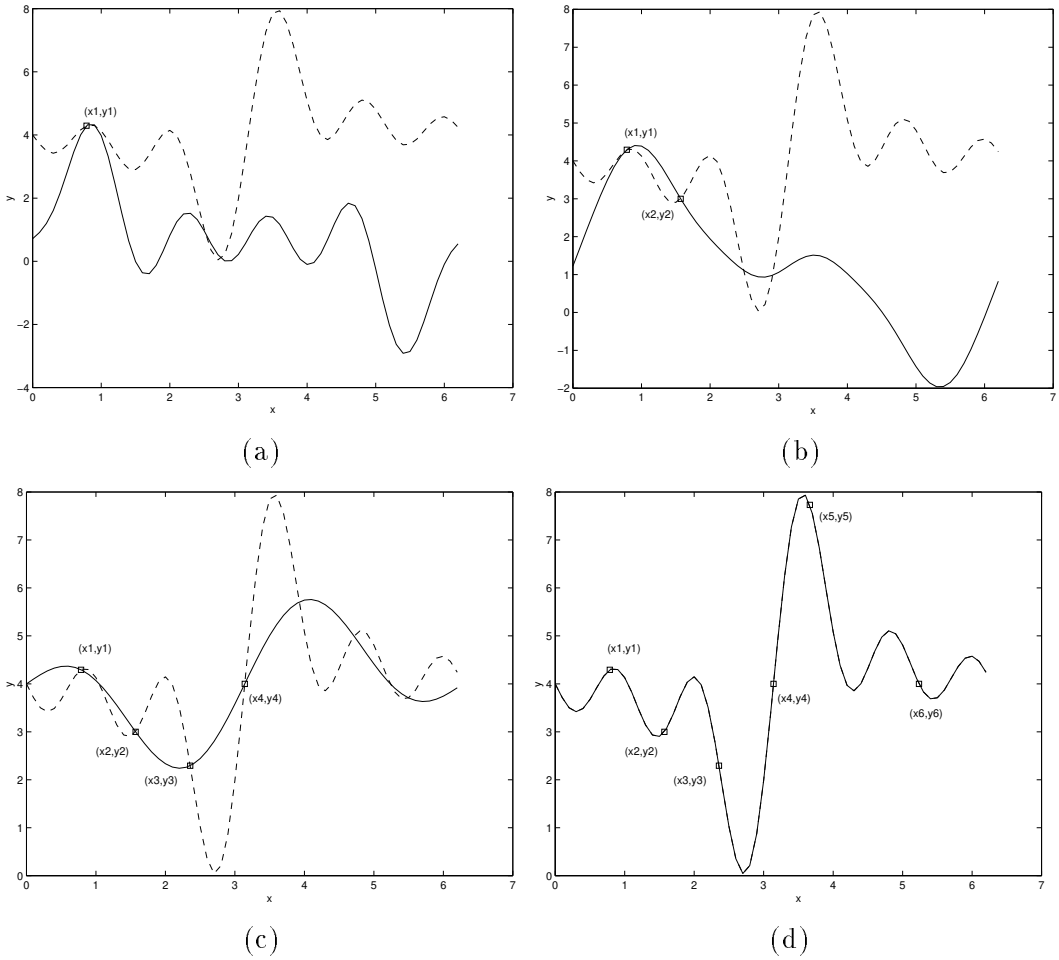


Figure 4.8: Incremental function approximation using the projection learning with (a)one (b)two (c)four and (d)six training data. The original function is shown by a dotted line and the learned function by a solid line. The training data is marked by tiny squares.

{Data #2}

Let us consider the second data point $(x_2, y_2) = (\frac{\pi}{2}, 3)$. The second sampling function $\psi_2(x)$ and it's projection onto the $\mathcal{R}(A^*)$, represented as $\tilde{\psi}_2(x)$, is computed as

$$\psi_2(x) = 1 + 2sin(x) - 2sin(3x) + 2sin(5x), \tag{4.74}$$

$$\begin{aligned}\tilde{\psi}_2(x) = \frac{1}{6}\{&5 + \sqrt{2} + (14 - \sqrt{2})sin(x) + (2\sqrt{2} - 2)sin(2x) \\ &-(10 + \sqrt{2})sin(3x) + (10 + \sqrt{2})sin(5x)\},\end{aligned} \tag{4.75}$$

and the corresponding learned function $f_2(x)$ is computed as

$$f_2(x) = 1.2210 + 2.0037sin(x) + 1.4963sin(2x) + 0.1123sin(3x) - 0.1125sin(5x). \tag{4.76}$$

The learned result is plotted in Fig.4.8(b). The generalization error turns out to be

$$\|f_2 - f\| = 3.7242 \tag{4.77}$$

©Sethu Vijayakumar 1998

which is slightly better than the case of f_1. As can be seen, the learned function correctly responds to the training data.

{Data #3, #4}
For brevity, we now just provide the data points and the newly learned functions along with the generalization error. After including data points $(x_3, y_3) = (\frac{3\pi}{4}, 3 - \frac{1}{\sqrt{2}})$ and $(x_4, y_4) = (\pi, 4)$, the learned function $f_4(x)$ is given as

$$f_4(x) = 4 - sin(x) + sin(2x) \tag{4.78}$$

and the generalization error

$$\|f_4 - f\| = 1.2247. \tag{4.79}$$

is considerably smaller than the previous cases. The learned result is plotted in Fig.4.8(c).

{Data #5, #6}
Finally, after incorporating $(x_5, y_5) = (\frac{5\pi}{3}, 4)$ and $(x_6, y_6) = (\frac{7\pi}{6}, 6+\sqrt{3})$, the learned function $f_6(x)$ exactly coincides with the original function f, in other words, there is no generalization error. This is shown in Fig.4.8(d).

Hence, threading through this concrete example, we saw how the exact incremental learning theory was applied in a toy example to illustrate the process of learning with additional training data. The functions were learnt by adding an extra correction component to the previously computed results and the generalization error was seen to decrease monotonically as was expected in case of noiseless training data.

4.10.2 The effect of training data location

In the previous experiment, we have seen that with increase in the number of training data, the learned function becomes closer to the original function. However, this is not always the case. The selection of the location of the training data has a huge effect on the generalizing ability of the newly learned function.

Continuing with the same example as in the last experiment, let us take up from the stage when we have learned using four training data $(x_1, y_1) = (\frac{\pi}{4}, 5 - \frac{1}{\sqrt{2}})$, $(x_2, y_2) = (\frac{\pi}{2}, 3)$, $(x_3, y_3) = (\frac{3\pi}{4}, 3 - \frac{1}{\sqrt{2}})$ and $(x_4, y_4) = (\pi, 4)$. The result of the learning is that we obtain the learned function f_4 given as

$$f_4(x) = 4 - sin(x) + sin(2x). \tag{4.80}$$

Now, let us consider the case of selecting the fifth data point.

{Case 1: $(x_5, y_5) \equiv (\frac{5\pi}{4}, 5 + \frac{1}{\sqrt{2}})$**}**
Under this selection of the training data, if we compute the value of t_m (refer

©Sethu Vijayakumar 1998

eq.(4.27)) and α(refer eq.(4.28)), we see that

$$t_m = (0, 0, -1, 2)^T \tag{4.81}$$

$$\alpha = 0 \tag{4.82}$$

Correspondingly, we see that

$$\tilde{\psi_5}(x) = 0 \tag{4.83}$$

$$f_5(x) = f_4(x). \tag{4.84}$$

This is reflected in Fig.4.9(a). The incorporation of this new data does not improve generalization and the learned function due to five training points (solid line) coincides with the learned function due to four points(dot-dashed). The generalization error remains at

$$\|f_5 - f\| = \|f_4 - f\| = 1.2247. \tag{4.85}$$

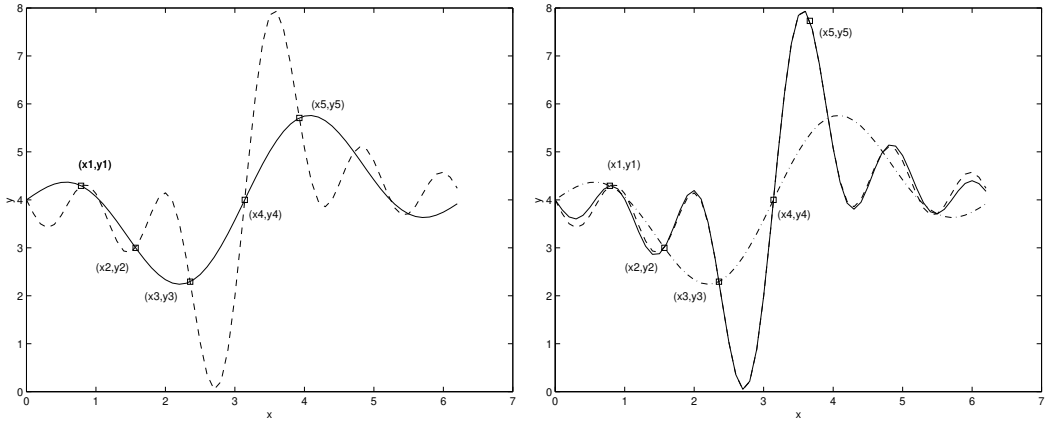


Figure 4.9: Effect of training data location on incremental learning results. The original function f(dashed), the function learned with four data points f_4(dot-dashed) and the function learned with five data points f_5(solid) are plotted. (a) shows learning with fifth data point selected in a redundant fashion; here f_4 (dot-dashed line) and f_5 (solid line) coincide (b) shows results with fifth data point chosen optimally.

{**Case 2:** $(x_5, y_5) \equiv (\frac{7\pi}{6}, 6 + \sqrt{3})$}

When we consider this alternate training data as the newly added fifth point, the corresponding computations yield

$$t_m = (0.1678, -0.1250, -0.6982, 1.6553)^T \tag{4.86}$$

$$\alpha = 3.7500 \tag{4.87}$$

and

$$\tilde{\psi_5}(x) = -1.5sin(3x) + \sqrt{3}sin(4x) - 1.5sin(5x) \tag{4.88}$$

$$f_5(x) = 4 - sin(x) + sin(2x) - 0.9464sin(3x) + 1.0928sin(4x) - 0.9464sin(5x). \tag{4.89}$$

©Sethu Vijayakumar 1998

As a result, the learned function becomes very close to the original function as can be seen from Fig.4.9(b) and the resultant generalization error is computed as

$$\|f_5 - f\| = 0.0847, \tag{4.90}$$

which is considerably smaller than the previous case. This example depicts how selection of training data is an important problem in the framework of learning. This problem will be handled in a more rigorous way in Chapter 6.

4.10.3 The model selection problem

In this section, we will try to illustrate the point made in Chapter 3 regarding the importance of resolving the bias-variance dilemma in the context of model selection. Here, we will consider an artificial example in which we take the problem of approximating a trigonometric function f given by

$$f = -2sin(x) + 3cos(x) + sin(2x) + 4cos(2x) - 0.5sin(3x) \tag{4.91}$$

using the Wiener optimization criterion. This function is represented by the solid line in Fig.4.10(a) and (b). To show the effect of the model selection, we consider two Hilbert spaces H_1 and H_2, the former being the subspace of the latter. Let H_1 be a 6-dimensional function space spanned by the orthonormal functions

$$\{\varphi_i^{(1)}(x)\}_{i=1}^{6} \equiv \{sin(x), cos(x), sin(2x), cos(2x), sin(3x), cos(3x)\},$$

where the inner product in H_1 is defined as

$$(f,g) = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)g(x)dx \;; \quad \forall f,g \in H_1. \tag{4.92}$$

In general, for function spaces of the kind spanned by orthonormal functions $\{sin(nx), cos(nx)\}_{n=1}^{N}$, the reproducing kernel can be written as

$$\begin{aligned} K(x,x') &= \sum_{n=1}^{N}(cos(nx)cos(nx') + sin(nx)sin(nx')) \\ &= \sum_{n=1}^{N} cos(n(x-x')) \\ &= \begin{cases} N & \text{if } x = x' \\ \frac{1}{2}[sin\frac{(2N+1)(x-x')}{2}/sin\frac{x-x'}{2} - 1] & \text{otherwise} \end{cases} \end{aligned} \tag{4.93}$$

Using the above result of eq.(4.93), the reproducing kernel for the Hilbert space H_1 can be written as

$$K^{(1)}(x,x') = \begin{cases} 3 & \text{if } x = x' \\ \frac{1}{2}[sin\frac{7(x-x')}{2}/sin\frac{x-x'}{2} - 1] & \text{otherwise} \end{cases} . \tag{4.94}$$

©Sethu Vijayakumar 1998

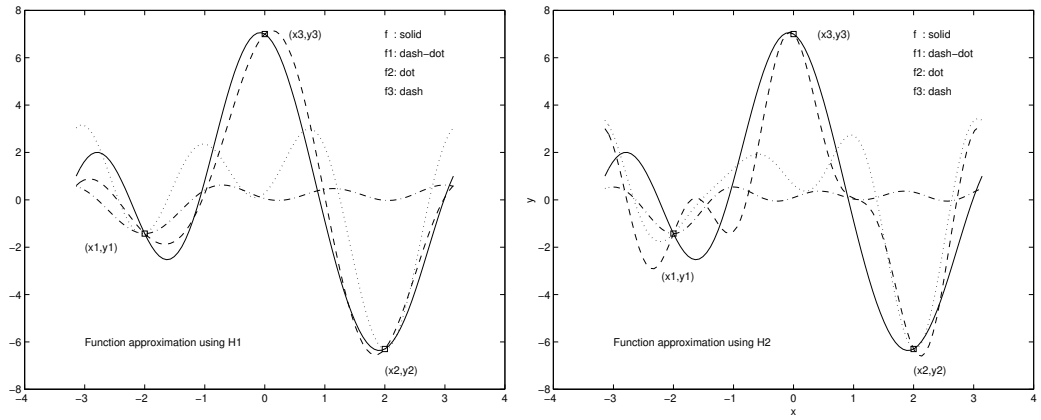
Similarly, consider H_2 to be an 8-dimensional function space spanned by the orthonormal functions

$$\{\varphi_n^{(2)}(x)\}_{n=1}^{8} \equiv \{sin(x), cos(x), sin(2x), cos(2x), sin(3x), cos(3x), sin(4x), cos(4x)\}$$

. The reproducing kernel $K_m^{(2)}(x)$ for H_2, by similar computation, works out to be

$$K^{(2)}(x, x') = \begin{cases} 4 & \text{if } x = x' \\ \frac{1}{2}[sin\frac{9(x-x')}{2}/sin\frac{x-x'}{2} - 1] & \text{otherwise} \end{cases} . \qquad (4.95)$$

Noiseless training data is obtained from the original function at locations $x_1 = -2$, $x_2 = 2$ and $x_3 = 0$ and is marked by tiny squares on the graphs. Here, for the sake of simplicity, we assume R, the correlation operator, to be an identity operator; in other words, all possible functions have an equal and independent probability of occurance.



(a) (b)

Figure 4.10: Function approximation result for original function $f = -2sin(x) + 3cos(x) + sin(2x) + 4cos(2x) - 0.5sin(3x)$ with one (f_1), two (f_2) and three (f_3) sample points. (a) uses a 6-dimensional Hilbert space H_1 while (b) uses an 8-dimensional Hilbert space H_2

Fig.4.10(a) shows the incremental learning results when learning with Hilbert space H_1. Here, f_1 refers to learned function using one sample point, f_2 with two sample points and so on. We can observe the improved generalization as the number of sample points increase. On the other hand, Fig.4.10(b) shows the approximation results for the same problem when we change the original search space to H_2. We observe, especially for f_3, that the generalization capability drops when we use a more general search space with more parameters (as in H_2) compared to one with a stronger and correct bias (as in H_1).

Hence, through these simulations, we have verified the effectiveness of using the exact incremental learning results to incorporate additional training data. It has also become evident that the selection of the location of training data is very important

©Sethu Vijayakumar 1998

for optimal generalization. Moreover, as against the previous belief of using a very general 'unbiased' solution search space, it has been shown that using a 'correctly biased' restricted search space leads to better generalization under the condition of finite (limited) training data. Hence, designing or searching for the right bias is an equally important problem as that of understanding learning per se.

©Sethu Vijayakumar 1998

Appendix A4

A4.1. Proof of Lemma 4.3: Alternate condition for Wiener learning

($\Rightarrow$)
Let the operator X satisfy the Wiener criterion. Then, from eq.(2.19) it follows that

$$XARA^* = RA^*. \tag{4.96}$$

For all $f \in \mathcal{R}(RA^*)$, there exists f' such that $f = RA^*f'$. Hence,

$$XAf = XARA^*f'. \tag{4.97}$$

Using eq.(4.96) in eq.(4.97), we have

$$XAf = RA^*f' = f. \tag{4.98}$$

($\Leftarrow$)
Assume for all $f \in \mathcal{R}(RA^*)$,

$$XAf = f. \tag{4.99}$$

Since this condition is for $f \in \mathcal{R}(RA^*)$, we can write that the following equation

$$XARA^*f' = RA^*f' \tag{4.100}$$

holds for all f', where $f = RA^*f'$. Hence, the lemma is proved

A4.2. Proof of Lemma 4.9

Since V is expressed as

$$V = \sum_{p,,q=1}^{m} v_{pq}(e_p^{(m)} \otimes \overline{e_q^{(m)}}),$$

it follows from eq.(4.23) that

$$\begin{aligned} \Gamma V \Gamma^* &= \Gamma[\sum_{p,q=1}^{m} v_{pq}(e_p^{(m)} \otimes \overline{e_q^{(m)}})]\Gamma^*, \\ &= \sum_{p,q=1}^{m} v_{pq}[(\Gamma e_p^{(m)} \otimes \overline{\Gamma e_q^{(m)}})], \\ &= \sum_{p,q=1}^{m} v_{pq}(e_p^{(m+1)} \otimes \overline{e_q^{(m+1)}}). \end{aligned}$$

This implies the lemma.

©Sethu Vijayakumar 1998

A4.3. Proof of Lemma 4.11

Since U_{m+1} is a non-negative matrix and has the structure represented by eq.(4.30), it can be decomposed as

$$U_{m+1} = \begin{pmatrix} U_m & s_m \\ s_m^T & \sigma \end{pmatrix} = \begin{pmatrix} G \\ g^T \end{pmatrix} \begin{pmatrix} G \\ g^T \end{pmatrix}^T, \tag{4.101}$$

where G is an $m \times m$ matrix and g is an m-dimensional column vector. It follows from the structure of eq.(4.30) and eq.(4.101) that

$$Gg = s_m. \tag{4.102}$$

Solving eq.(4.102) for g, we have for some u

$$g = G^\dagger s_m + Cu, \tag{4.103}$$

where C is defined as

$$C = I - G^\dagger G. \tag{4.104}$$

Since $Cu \perp G^\dagger s_m$, eqs.(4.30), (4.101), and (4.103), we have

$$\sigma = \|g\|^2 = \|G^\dagger s_m\|^2 + \|Cu\|^2. \tag{4.105}$$

We also have

$$Cg = C(G^\dagger s_m + Cu) = C^2 u = Cu. \tag{4.106}$$

Hence, from eqs.(4.105) and (4.106)

$$\begin{aligned} \|Cg\|^2 &= \sigma - \|G^\dagger s_m\|^2 \\ &= \sigma - ((GG^T)^\dagger s_m, s_m) \\ &= \sigma - (U_m^\dagger s_m, s_m), \end{aligned} \tag{4.107}$$

since $U_m = GG^T$. Hence, eqs.(4.27) and (4.28) yield

$$\|Cg\|^2 = \alpha. \tag{4.108}$$

From eqs.(4.108) and (4.104), we have

$$\alpha = \|(I - GG^T)g\|^2. \tag{4.109}$$

Hence, $\alpha = 0$ if and only if $(I - GG^T)g = 0$. In other words, $\alpha = 0$ if and only if $g \in \mathcal{R}(G^T)$. Therefore, $\alpha = 0$ if and only if

$$rank(\begin{pmatrix} G \\ g^T \end{pmatrix}^T) = rank(G). \tag{4.110}$$

©Sethu Vijayakumar 1998

On the other hand, from eq.(4.101), we have

$$rank(U_m) = rank(G) \text{ and} \tag{4.111}$$

$$rank(U_{m+1}) = rank(\begin{pmatrix} G \\ g^T \end{pmatrix}^T). \tag{4.112}$$

Looking at eqs.(4.110), (4.111) and (4.112), we conclude that $\alpha = 0$ if and only if

$$rank(U_{m+1}) = rank(U_m). \tag{4.113}$$

Since $\alpha \geq 0$ because of Lemma 4.2, eq.(4.113) implies Lemma 4.11.

A4.4. Proof of Lemma 4.14

From eqs.(4.28) and (4.26), we have

$$\alpha = (R\psi_{m+1})(x_{m+1}) - (t_m, s_m). \tag{4.114}$$

Looking at the second term of the expansion in eq.(4.114) and using eqs.(4.25) and (4.27), we have

$$\begin{aligned} (t_m, s_m) &= (U_m^\dagger A_m R\psi_{m+1}, A_m R\psi_{m+1}) \\ &= (RA_m^* U_m^\dagger A_m R\psi_{m+1}, \psi_{m+1}). \end{aligned} \tag{4.115}$$

Using eq.(4.20) and Theorem 4.1, we can write eq.(4.115) as

$$\begin{aligned} (t_m, s_m) &= (X_m A_m R\psi_{m+1}, \psi_{m+1}) \\ &= (P_{S_m, S_{mc}} R\psi_{m+1}, \psi_{m+1}) \\ &= (P_{S_m, S_{mc}} R\psi_{m+1})(x_{m+1}). \end{aligned} \tag{4.116}$$

Substituting eq.(4.116) in eq.(4.114) and using the definition of ϕ,ϕ_m and ϕ_{mc}, we have

$$\begin{aligned} \alpha &= (R\psi_{m+1})(x_{m+1}) - (P_{S_m, S_{mc}} R\psi_{m+1})(x_{m+1}) \\ &= \phi(x_{m+1}) - P_{S_m, S_{mc}} \phi(x_{m+1}) \\ &= \phi_{mc}(x_{m+1}). \end{aligned} \tag{4.117}$$

A4.5: Proof of Theorem 4.4

From eqs.(4.38) and (4.27),

$$\begin{aligned} v_m &= \frac{1}{\alpha}(R\psi_{m+1} - RA_m^* t_m) \\ &= \frac{1}{\alpha}(R\psi_{m+1} - RA_m^* U_m^\dagger s_m). \end{aligned} \tag{4.118}$$

©Sethu Vijayakumar 1998

Applying eqs.(4.20) and (4.25) to eq.(4.118), we have

$$\begin{aligned} v_m &= \frac{1}{\alpha}(R\psi_{m+1} - X_m s_m) \\ &= \frac{1}{\alpha}(R\psi_{m+1} - X_m A_m R\psi_{m+1}) \\ &= \frac{1}{\alpha}(I - X_m A_m)R\psi_{m+1}. \end{aligned} \tag{4.119}$$

Using the result of Theorem 4.1 and eq.(4.50), we write eq.(4.119) as

$$\begin{aligned} v_m &= \frac{1}{\alpha}(I - P_{S_m,S_{mc}})R\psi_{m+1} \\ &= \frac{1}{\alpha}P_{S_{mc},S_m}R\psi_{m+1} \\ &= \frac{1}{\alpha}P_{S_{mc},S_m}\phi. \end{aligned} \tag{4.120}$$

By using definition of ϕ_{mc} and eq.(4.51), we can reduce eq.(4.120) to

$$v_m = \frac{\phi_{mc}}{\phi_{mc}(x_{m+1})}. \tag{4.121}$$

On the other hand, on substituting the value of Y_m from eq.(4.39) into eq.(4.48), we have

$$\begin{aligned} f_{m+1} &= f_m - (v_m \otimes \overline{t_m})y^{(m)} + f(x_{m+1})v_m \\ &= f_m - (y^{(m)}, t_m)v_m + f(x_{m+1})v_m \\ &= f_m + [f(x_{m+1}) - (y^{(m)}, t_m)]v_m. \end{aligned} \tag{4.122}$$

Now, let us look at the term $(y^{(m)}, t_m)$ in the above equation. It can be written as

$$(y^{(m)}, t_m) = (A_m f, t_m) = (f, A_m^* t_m). \tag{4.123}$$

Assume $f \in \mathcal{R}(R)$, which holds true for Wiener learning. Then, $f = Ru$ for some u. Hence, eq.(4.123) can be written as

$$(y^{(m)}, t_m) = (Ru, A_m^* t_m) = (u, RA_m^* t_m). \tag{4.124}$$

Expanding for t_m using eqs.(4.27) and (4.25) in eq.(4.124) and then, using eq.(4.20), we have

$$\begin{aligned} (y^{(m)}, t_m) &= (u, RA_m^* U_m^\dagger A_m R\psi_{m+1}) \\ &= (u, X_m A_m R\psi_{m+1}). \end{aligned} \tag{4.125}$$

Using Theorem 4.1, eq.(4.125) can be written as

$$\begin{aligned} (y^{(m)}, t_m) &= (u, P_{S_m,S_{mc}}R\psi_{m+1}) \\ &= (P_{S_m,S_{mc}}u, R\psi_{m+1}) \\ &= (RP_{S_m,S_{mc}}u, \psi_{m+1}), \end{aligned} \tag{4.126}$$

©Sethu Vijayakumar 1998

It can be easily shown that $P_{S_m,S_{mc}}R = RP_{S_m,S_{mc}}$. Using this result in eq.(4.126), we have

$$\begin{aligned}(y^{(m)},t_m) &= (P_{S_m,S_{mc}}Ru,\psi_{m+1}) \\ &= (P_{S_m,S_{mc}}f,\psi_{m+1}) \\ &= (f_m,\psi_{m+1}). \end{aligned} \tag{4.127}$$

Using the property of the function ψ_{m+1}, eq.(4.127) becomes

$$(y^{(m)},t_m) = f_m(x_{m+1}). \tag{4.128}$$

Now, using eqs.(4.121) and (4.128) in eq.(4.122), we have

$$\begin{aligned}f_{m+1} &= f_m + [f(x_{m+1}) - f_m(x_{m+1})]\frac{\phi_{mc}}{\phi_{mc}(x_{m+1})} \\ &= f_m + \frac{f_{mc}(x_{m+1})}{\phi_{mc}(x_{m+1})}\phi_{mc}. \end{aligned} \tag{4.129}$$

©Sethu Vijayakumar 1998

5

Learning sensorimotor maps using apriori knowledge

5.1 Introduction

In earlier chapters, we have laid down a framework for performing exact incremental learning and devised a specific method of carrying out learning using only the newly available information and the results of learning carried out so far. Empirical evaluations of the function approximation capabilities were carried out using artificially simulated examples.

Here, we will try to apply the exact incremental learning framework to a real world problem of learning the sensorimotor map of a robot arm. Traditional techniques of learning the sensorimotor map, a relation mapping the desired angular position, velocities and accelerations to the appropriate torques at each joints, has been based on parametric estimation. The idealized rigid body physical models were employed and the remaining variables or parameters were estimated with linear regression. However, due to the inaccuracies of estimating the physical parameters like the link length, mass, center of gravity etc. most of the inverse dynamics model used in control were a rough approximation to the actual dynamics of the system. Control was done using a feedback error correction method in which the errors in the trajectory due to inaccurate dynamics model was compensated by a scaled error signal of the difference between the actual and desired trajectory. Due to this, the speed of the movement (determined by the maximum gain possible in a PID type controller) is limited.

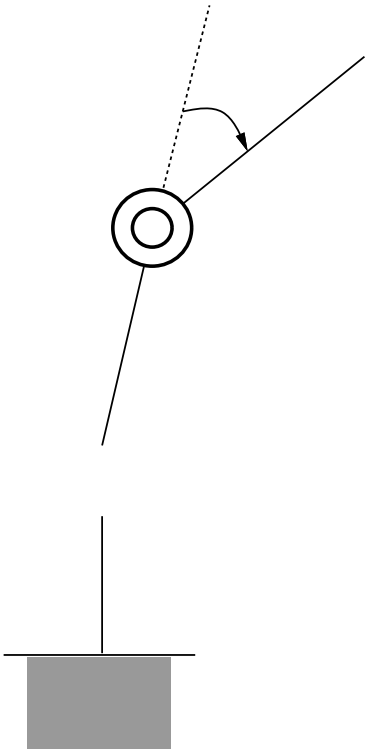


Figure 5.1: An idealized model of a 2 DOF robot arm

We propose to use a method for estimating the inverse dynamics which does not directly require the use of the physical parameters of the model. The exact incremental learning framework is well suited for this purpose and approximates the inverse dynamics function without the use of physical parameters. We will use the projection learning criterion discussed in Section 2.3.4 as the optimization criterion for the problem. To apply this learning framework, we have to decide on the Hilbert space to search for, i.e. decide on the original function space H. Many different approaches including techniques described in Chapter 3 can be employed. However, here we will make use of the apriori knowledge of the equations of dynamics which is known for the 2 DOF robot arm to construct our search space and evaluate the approximation results based on this technique.

5.2 Equations of dynamics of motion

In this chapter, we will work with a two-joint arm (2 DOF) robot, an idealized model of which is shown in Fig. 5.1. The 2 DOF robot has 2 links l_1 and l_2 and each of link can move in an angular fashion to modify the angle it makes with the base link,

©Sethu Vijayakumar 1998

θ_1 and θ_2, respectively. The joint between the base and the first link is referred to as the *shoulder joint* and the one between link one and link two is called the *elbow joint*. The end of the manipulator is at a position $r = (x, y, z)$, this representing the coordinates in the work space. The state of the robot arm at any point in time is specified completely by the state variables $(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})$, where $\boldsymbol{\theta} \equiv (\theta_1\, \theta_2)$ represents the angular position, $\dot{\boldsymbol{\theta}} \equiv (\dot{\theta_1}\, \dot{\theta_2})$ represents the angular velocity and $\ddot{\boldsymbol{\theta}} \equiv (\ddot{\theta_1}\, \ddot{\theta_2})$ represents the angular acceleration of each link. We can move the robot arm by applying torques τ_1 and τ_2 to each of the two joints, respectively. We will denote the consolidated torque by a vector $\boldsymbol{\tau} \equiv (\tau_1\, \tau_2)$.

Definition 5.1 (Robot control) *The problem of robot control, stated in it's simplest form, is to move the manipulator end along a particular trajectory at a given velocity and acceleration by applying the appropriate torques to the joints.*

Next, we define some of the transformation or mappings that are used in order to control the movement of the robot arm.

Forward Dynamics The *forward dynamics* of the model gives us a relationship about how the state of the robot changes when a particular torque is applied. If we represent the mapping by g, it provides the relationship

$$\ddot{\boldsymbol{\theta}} = g(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\tau}). \tag{5.1}$$

Inverse Dynamics The *inverse dynamics* is the relationship which provides what torque, $\boldsymbol{\tau}$, should be applied to each of the joints so that the arm moves with a particular desired angular velocity $\dot{\boldsymbol{\theta}}_d$ and desired acceleration $\ddot{\boldsymbol{\theta}}_d$ for a particular desired displacement $\boldsymbol{\theta}_d$. It is this relationship which will be useful in the robot arm control and is represented by a mapping f such that

$$\boldsymbol{\tau} = f(\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d). \tag{5.2}$$

This mapping is also referred to as the *sensorimotor map* since it provides the torques or motor commands corresponding to a desired state which is perceived through a sensory action. It is this mapping that we will attempt to approximate from training samples using the exact incremental learning method devised here.

Inverse Kinematics *Inverse kinematics* is the mapping which relates the work space coordinates of the manipulator tip $\mathbf{r} \equiv (x, y, z)$ to the internal angular coordinates of the robot joints $\boldsymbol{\theta} \equiv (\theta_1, \theta_2)$. In this work, we will not deal with the workspace coordinates but rather with the joint angles and assume that the inverse kinematics has already been acquired or learned.

©Sethu Vijayakumar 1998

Based on the rigid body dynamics approximation and using the idealized model of a 2 DOF robot arm, we can approximate the dynamics equations of motion as follows. Let I_i represent the moment of inertia of link i and L_{G_i} the distance between the joint i and the center of mass of link i. Furthermore, let L_i and m_i represent the length and mass of the link i, respectively. Let g be the gravitational constant. Then, the respective torques τ_1 and τ_2 required for each of the two joints to achieve a given state can be given as

$$\begin{aligned}
\tau_1 &= (I_1 + m_1 {L_{G1}}^2 + I_2 + m_2 {L_{G2}}^2 + m_2 L_1^2)\,\ddot{\theta_1} \\
&\quad + (I_2 + m_2 {L_{G2}}^2)\,\ddot{\theta_2} \\
&\quad + 2\, m_2 L_1 L_{G2}\,\ddot{\theta_1}\cos\theta_2 \\
&\quad + m_2 L_1 L_{G2}\,\ddot{\theta_2}\cos\theta_2 \\
&\quad - m_1 L_1 L_{G2}\,{\dot{\theta_2}}^2 \sin\theta_2 \\
&\quad - m_1 L_1 L_{G2}\,\dot{\theta_1}\,\dot{\theta_2}\sin\theta_2 \\
&\quad - (m_1 L_{G1} + m_2 L_1)\, g \sin\theta_1 \\
&\quad - m_2 L_{G2}\, g \sin\theta_1 \cos\theta_2 \\
&\quad - m_2 L_{G2}\, g \sin\theta_2 \cos\theta_1 \qquad (5.3) \\
\tau_2 &= (I_2 + m_2 {L_{G2}}^2)\,\ddot{\theta_1} \\
&\quad + m_2 L_1 L_{G2}\,\ddot{\theta_1} cos\theta_2 \\
&\quad + (I_2 + m_2 {L_{G2}}^2)\,\ddot{\theta_2} \\
&\quad + m_2 L_1 L_{G2}\,\dot{\theta_2}\sin\theta_2 \\
&\quad - m_2 L_{G2}\, g \sin\theta_1 \cos\theta_2 \\
&\quad - m_2 L_{G2}\, g \sin\theta_2 \cos\theta_1 \qquad (5.4)
\end{aligned}$$

The computation of the above relationship can be derived from first principles based on the dynamics of motion of an idealized model. The details of this derivation is shown in Appendix A5.

The computation of the dynamics has not taken into account the effect of friction and environmental factors and is under the assumption of the simplification of the arm model to a rigid body approximation. We will use this physical model as an apriori information to build a suitable Hilbert function space, as will be detailed out in the next section.

©Sethu Vijayakumar 1998

5.3 Hilbert search space for 2-DOF robot arm

The ultimate aim of the current learning problem is to learn the sensorimotor map, f given in eq.(5.2), which corresponds to the torques necessary at each joint for moving the arm with a particular velocity and acceleration to a particular point. Here, we will deal with the problem of learning this multidimensional output map (here, two dimensional) by separately learning two different functions, each corresponding to one of the output dimensions as

$$\tau_1 = f_1(\theta_1, \dot{\theta_1}, \ddot{\theta_1}, \theta_2, \dot{\theta_2}, \ddot{\theta_2}), \tag{5.5}$$

$$\tau_2 = f_2(\theta_1, \dot{\theta_1}, \ddot{\theta_1}, \theta_2, \dot{\theta_2}, \ddot{\theta_2}). \tag{5.6}$$

In order to learn these functions, we have to decide on a Hilbert search space that includes them. Naively, without the use of any apriori knowledge and looking at all possible combinations of the variables (including the sines and cosines of θ_1 and θ_2), we have a regression on a space whose dimensionality is given as

$$dim(H) \leq \sum_{n=1}^{10} \left(\begin{array}{c} 10 \\ n \end{array} \right) = 2^{10} - 1. \tag{5.7}$$

This is an extremely large space and requires a tremendous amount of data to get reasonable regression results. However, here we use the model based on rigid dynamics as apriori knowledge, and construct the Hilbert spaces H_1 and H_2 which includes function f_1 and f_2, respectively. Concretely, H_1 is the space spanned by the family of functions in eq.(5.3) and H_2 is the space spanned by the family of functions in eq.(5.4), respectively.

Let $[-a,a]$, $[-b,b]$, $[-c,c]$, $[-d,d]$ represent the constrained range of the variables $\dot{\theta_1}, \dot{\theta_2}, \ddot{\theta_1}, \ddot{\theta_2}$, respectively. For any two functions $f, g \in H_1$ or $f, g \in H_2$, we define the inner product as

$$(f,\ g) = \frac{1}{W} \int_{-d}^{d} \int_{-c}^{c} \int_{-b}^{b} \int_{-a}^{a} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} f\, \overline{g}\, d\theta_1 d\theta_2 d\dot{\theta_1} d\dot{\theta_2} d\ddot{\theta_1} d\ddot{\theta_2}, \tag{5.8}$$

where $W = 64\pi^2 abcd$. Based on this, we compute the orthonormal basis (ONB) of the space H_1 and H_2. The ONB, $\{\phi_n^{(1)}\}_{n=1}^{9}$, for H_1 are given as

$$\phi_1^{(1)} = \frac{\sqrt{3}}{c}\, \ddot{\theta_1}, \tag{5.9}$$

$$\phi_2^{(1)} = \frac{\sqrt{6}}{c}\, \ddot{\theta_1} \cos\theta_2, \tag{5.10}$$

$$\phi_3^{(1)} = \frac{\sqrt{3}}{d}\, \ddot{\theta_2}, \tag{5.11}$$

$$\phi_4^{(1)} = \frac{\sqrt{6}}{d}\, \ddot{\theta_2} \cos\theta_2, \tag{5.12}$$

©Sethu Vijayakumar 1998

$$\phi_5^{(1)} = \frac{\sqrt{10}}{b^2}\,\dot{\theta_2}^{\,2}\sin\theta_2, \tag{5.13}$$

$$\phi_6^{(1)} = \frac{\sqrt{18}}{ab}\,\dot{\theta_1}\dot{\theta_2}\sin\theta_2, \tag{5.14}$$

$$\phi_7^{(1)} = \sqrt{2}\sin\theta_1, \tag{5.15}$$

$$\phi_8^{(1)} = 2\sin\theta_1\cos\theta_2, \tag{5.16}$$

$$\phi_9^{(1)} = 2\sin\theta_2\cos\theta_1. \tag{5.17}$$

Similarly, the ONB, $\{\phi_n^{(2)}\}_{n=1}^{6}$, for H_2 are given as

$$\phi_1^{(2)} = \frac{\sqrt{3}}{c}\,\ddot{\theta_1}, \tag{5.18}$$

$$\phi_2^{(2)} = \frac{\sqrt{6}}{c}\,\ddot{\theta_1}\cos\theta_2, \tag{5.19}$$

$$\phi_3^{(2)} = \frac{\sqrt{3}}{d}\,\ddot{\theta_2}, \tag{5.20}$$

$$\phi_4^{(2)} = \frac{\sqrt{10}}{a^2}\,\dot{\theta_1}^{\,2}\sin\theta_2, \tag{5.21}$$

$$\phi_5^{(2)} = 2\sin\theta_1\cos\theta_2, \tag{5.22}$$

$$\phi_6^{(2)} = 2\sin\theta_2\cos\theta_1. \tag{5.23}$$

5.4 Experiments in learning of the sensorimotor map

In this section, we will use the Hilbert spaces H_1 and H_2 constructed in Section 5.3 to perform incremental learning using real 2DOF robot data received from the simulator, indigenously developed largely through efforts of Dr.Stefan Schaal.

Let us represent the sextuple input for the 2DOF robot arm by a consolidated vector $\mathbf{x}$ such that $\mathbf{x} = (\ \theta_1\ \dot{\theta_1}\ \ddot{\theta_1}\ \theta_2\ \dot{\theta_2}\ \ddot{\theta_2}\ )$ for the sake of convenience. Based on the ONBs constructed in the previous section, the reproducing kernels $K_1(\mathbf{x},\mathbf{x}')$ and $K_2(\mathbf{x},\mathbf{x}')$ for the Hilbert spaces H_1 and H_2, respectively can be computed as

$$K_i(\mathbf{x},\mathbf{x}') = \sum_{n=1}^{N} \phi_n^{(i)}(\mathbf{x})\overline{\phi_n^{(i)}(\mathbf{x}')}, \quad i = 1,2, \tag{5.24}$$

where N corresponds to the dimensionality of the Hilbert space H_i. Let us instantiate functions $\psi_m^{(i)}$ as

$$\psi_m^{(i)}(\mathbf{x}) \equiv K_i(\mathbf{x},\mathbf{x}_m), \quad i = 1,2, \tag{5.25}$$

using the m-th sample point $\mathbf{x}_m$. Then, the sampling operator $A_m^{(i)}$ using m sample points can be constructed, based on eq.(4.1), as

$$A_m^{(i)} = \sum_{n=1}^{m} (e_n^{(m)} \otimes \overline{\psi_n^{(i)}}), \quad i = 1,2. \tag{5.26}$$

©Sethu Vijayakumar 1998

Also, let us define $\psi_{mc}^{(i)}$ as

$$\psi_{mc}^{(i)} = P_{\mathcal{N}(A_m^{(i)})}\psi_m^{(i)}, \quad i = 1, 2. \tag{5.27}$$

We assume we have a set of training data $\{\mathbf{x}_m, \tau_{1,m}, \tau_{2,m}\}_{m=1}^{M}$. Since we are using projection learning, based on the incremental learning results of Table 4.2, we can compute the sensorimotor maps f_1 and f_2 corresponding to the shoulder and elbow joints (refer eqs.(5.5) and (5.6)) incrementally as

$$f_1^{m+1} = f_1^m + \frac{\tau_{1,m+1} - f_1^m(\mathbf{x}_{m+1})}{\psi_{mc}^{(1)}(\mathbf{x}_{m+1})}\psi_{mc}^{(1)}, \tag{5.28}$$

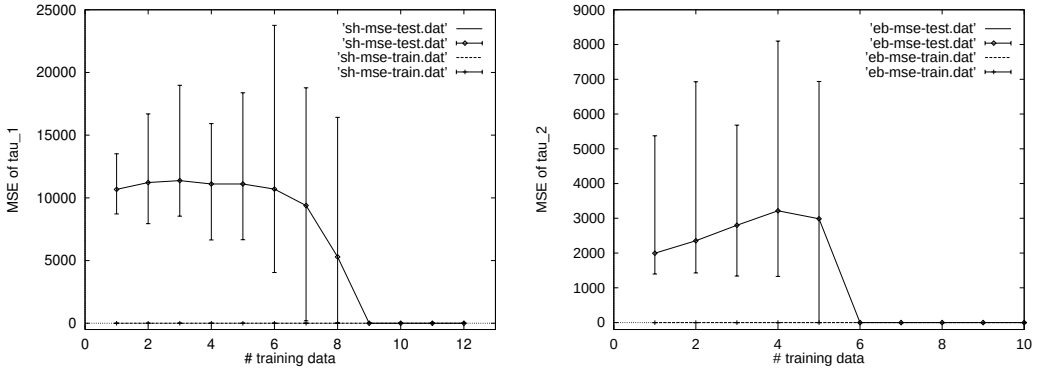
$$f_2^{m+1} = f_2^m + \frac{\tau_{2,m+1} - f_2^m(\mathbf{x}_{m+1})}{\psi_{mc}^{(2)}(\mathbf{x}_{m+1})}\psi_{mc}^{(2)}. \tag{5.29}$$

5.4.1 Empirical evaluation results and learning curves

Based on eqs.(5.28) and (5.29), we performed incremental learning using training data obtained from the simulator. In our analysis, we looked at training using different number of training data (#training data = 1,2,...,12) for the shoulder and elbow joints, respectively. Furthermore, to look at the effect of the selection of training data, we considered 100 different sets of training data. For each of these cases, the Mean Squared Error (MSE) on the training data and the MSE on a test data set (consisting of 100 randomly selected points) were computed. The mean MSE for a particular size of the training set and their maximum and minimum bounds are also computed. The true generalization error, computed as the norm of the difference between the learned function and the true function (which in this case, we know), is also computed for each of these cases. Summarizing the different cases for which the computations were carried out:

1. For SHOULDER joint, DO ...
   - For training set size #train_size = 1 to 12
     - Select random test set of size 100.
     - For K =1 to 100
       (a) Select training data set #K of size = #train_size.
       (b) Learn with training set #K.
       (c) Compute MSE on training data.
       (d) Compute MSE on test data.
       (e) Compute the true generalization error (GenErr).
     - End. (For K ..)

©Sethu Vijayakumar 1998



(a) Shoulder Torques (b) Elbow Torques

Figure 5.2: The Mean Squared Error (MSE) on the training data(dashed line) and on a set of 100 test data(solid line) for the sensorimotor map learned with different number of training data; the bars represent the maximum and minimum values and the dot represents the mean of the MSE.

- Compute max.MSE, min.MSE and mean.MSE for the test data and the training data.
- Compute max.GenErr, min.GenErr and mean.GenErr.

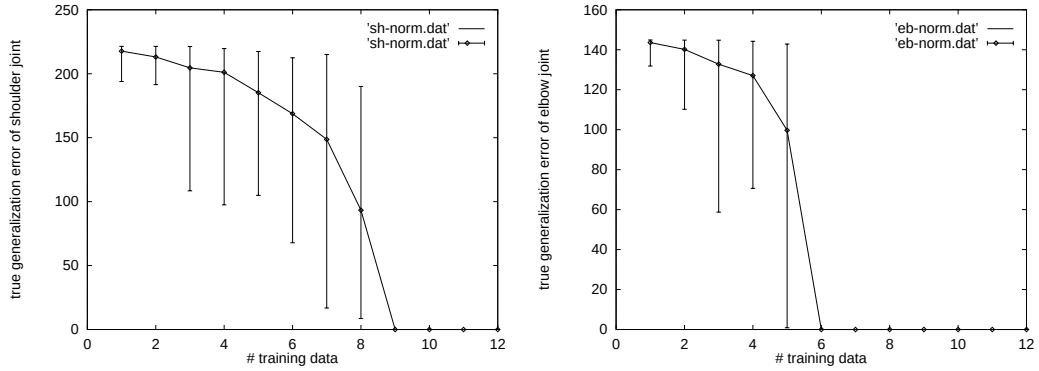
- End. (For training ..)

2. For ELBOW joint, DO ...

- For training set size #train_size = 1 to 12
  - Select random test set of size 100.
  - For K =1 to 100
    (a) Select training data set #K of size = #train_size.
    (b) Learn with training set #K.
    (c) Compute MSE on training data.
    (d) Compute MSE on test data.
    (e) Compute the true generalization error (GenErr).
  - End. (For K ..)
  - Compute max.MSE, min.MSE and mean.MSE for the test data and the training data.
  - Compute max.GenErr, min.GenErr and mean.GenErr.
- End. (For training ..)

The summarized result of the experiments are given in Figs.5.2 and 5.3. Fig.5.2(a) plots the average of the mean squared error (MSE) in the learning of the sensorimotor map at the shoulder joint using 100 different training sets of a particular size, the training set size ranging from 1 to 12. The graphs also shows the lower and upper

©Sethu Vijayakumar 1998



(a) Shoulder Torques (b) Elbow Torques

Figure 5.3: The actual generalization error (the norm of the difference between the true function and the learned function) for the sensorimotor map learned with different number of training data; the bars represent the maximum and minimum values and the dot represents the mean of the true error.

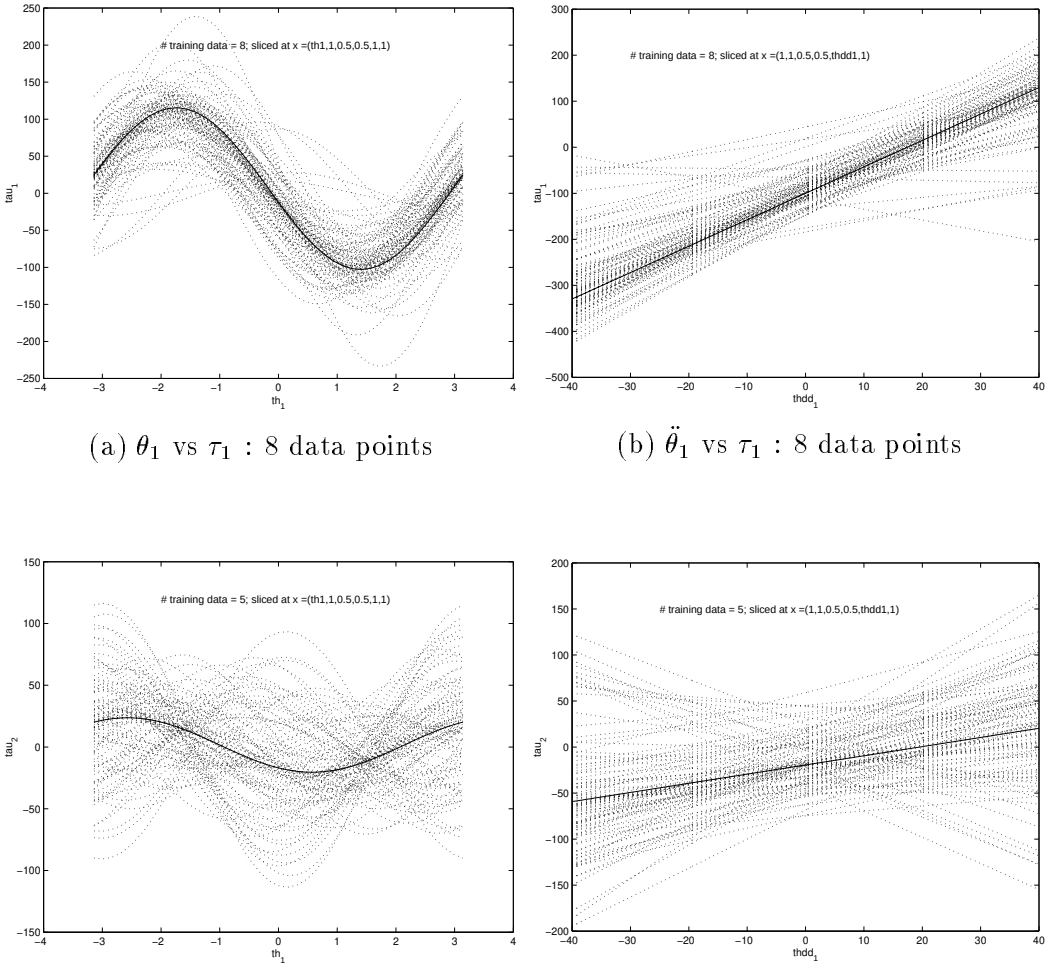
bound of the MSE for each case. Fig.5.2(b) plots the results of the similar analysis for the elbow joint. The training data error, plotted in both the graphs (a) and (b), is always zero because we are learning with a noise free model. It can be seen that the MSE decreases (on the average) and attains zero value with 9 training data for the shoulder torques and with 6 points for the elbow torques.

Fig.5.3 plots the real generalization error for the shoulder and elbow joints. Fig.5.3(a) depicts the monotonous fall in the generalization error with increase in training data till perfect learning is achieved with 9 data points. Similarly, Fig.5.3(b) shows the monotonous decrease of generalization error and finally, learns perfectly with only 6 training data. It is interesting to note that the true generalization error *decreases monotonously* with increased data points in the noiseless case. However, this is not the case for the MSE of a validation or test set consisting of 100 randomly chosen data points. We can notice an initial increase in the MSE on test set of the elbow torques with increase in number of training data in Fig.5.2(b) as compared to the monotonous decrease of the corresponding generalization error as shown in Fig.5.3(b). This comparison shows that the monitoring of error over a randomly chosen test or cross validation based methods have serious pitfalls when it comes to approximating the true generalization error in a learning process.

5.4.2 The variance of solution based on training data location

We found that for the sensorimotor map of the shoulder joint, training with 9 independent data points invariably provided perfect approximation results. Similarly, for the elbow joints, training with 6 independent points also resulted in perfect generalization. However, we saw that we got a fair degree of variance in solution when training with lesser number of training data than required for perfect learning. This

©Sethu Vijayakumar 1998



(a) θ_1 vs τ_1 : 8 data points

(b) $\ddot{\theta}_1$ vs τ_1 : 8 data points

(c) θ_1 vs τ_2 : 5 data points

(d) $\ddot{\theta}_1$ vs τ_2: 5 data points

Figure 5.4: Variance of the approximation results with location of the training data

suggests the importance of the location of the training data.

To visualize this effect, we performed training on the sensorimotor map of the shoulder joint with 8 training data points and plotted the various cross section of the multidimensional function. Fig.5.4(a) shows the cross section of learned result τ_1 when we sliced across $(\theta_1, \theta_2, \dot{\theta_1}, \dot{\theta_2}, \ddot{\theta_1}, \ddot{\theta_2}) = (\theta_1, 1, 0.5, 0.5, 1, 1)$; i.e., varying θ_1 while keeping all other variables constant. The true function at this cross section is plotted with a dark line while the 100 other functions learned with 100 variations of the training data set is plotted with dotted lines. Similarly, Fig.5.4(b) plots the results of the same analysis across another cross section, $(\theta_1, \theta_2, \dot{\theta_1}, \dot{\theta_2}, \ddot{\theta_1}, \ddot{\theta_2}) = (1, 1, 0.5, 0.5, \ddot{\theta_1}, 1)$; i.e., varying $\ddot{\theta_1}$. The variation in solution based on location of the training data is very evident. Similar plots for the elbow joint (τ_2) with cross section at $(\theta_1, \theta_2, \dot{\theta_1}, \dot{\theta_2}, \ddot{\theta_1}, \ddot{\theta_2}) = (\theta_1, 1, 0.5, 0.5, 1, 1)$ for learning with a training set size of 5 (Fig.5.4(c)) and with cross section at $(\theta_1, \theta_2, \dot{\theta_1}, \dot{\theta_2}, \ddot{\theta_1}, \ddot{\theta_2}) = (1, 1, 0.5, 0.5, \ddot{\theta_1}, 1)$ (Fig.5.4(d)) for the same case are shown.

©Sethu Vijayakumar 1998

This analysis shows the importance of the selection of the location of training data in the case when sufficient data is not available or for obtaining better generalization results with limited amount of training data. The next Chapter will deal with this problem of active data selection in a more rigorous manner.

Acknowledgment

I would like to thank Mr. Ryusuke Michigami for help in implementation of the ideas of sensorimotor learning in the 2 DOF robot arm and for compiling the simulation results in this chapter.

©Sethu Vijayakumar 1998

Appendix A5

A5.1. Basic vector algebra for robot dynamics

The outer product $\times$ is defined to possess the following properties. For unitary basis vectors of the internal coordinate representation, $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ and any given vectors $\boldsymbol{a}$,$\boldsymbol{b}$,$\boldsymbol{c}$, we have

1. $\boldsymbol{a} \times \boldsymbol{b} = -\boldsymbol{b} \times \boldsymbol{a}$
2. $\boldsymbol{a} \times (\boldsymbol{b} + \boldsymbol{c}) = \boldsymbol{a} \times \boldsymbol{b} + \boldsymbol{a} \times \boldsymbol{c}$
3. $\boldsymbol{i} \times \boldsymbol{j} = \boldsymbol{k} \qquad \boldsymbol{j} \times \boldsymbol{k} = \boldsymbol{i} \qquad \boldsymbol{k} \times \boldsymbol{i} = \boldsymbol{j}$
   $\boldsymbol{i} \times \boldsymbol{i} = \boldsymbol{j} \times \boldsymbol{j} = \boldsymbol{k} \times \boldsymbol{k} = 0$

An elementwise representation of the outer product looks as

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \times \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix}. \tag{5.30}$$

We can define rotationary motion in a body by looking at the rotation at a particular point (represented by a vector) $\boldsymbol{r}$. Then, the point $\boldsymbol{r}$ is defined to move with an angular velocity of $\boldsymbol{\omega}$. In general, for a vector $\boldsymbol{r}$ making a forward motion with angular rotation velocity of $\boldsymbol{\omega}$, the partial differential (d/dt) [ ] for a vector [ ] is defines as

$$\frac{d}{dt}[\,] = \dot{[\,]} = \dot{[\,]}^* + \boldsymbol{\omega} \times [\,]. \tag{5.31}$$

Here $\cdot$ with the $*$ on top denotes the differential of the non-angular part of the motion. Using this, we can define the the velocity $\boldsymbol{v}$ of the vector $\boldsymbol{r}$ making screw motion with angular velocity $\boldsymbol{\omega}$ as

$$\boldsymbol{v} = \dot{r} = \dot{\boldsymbol{r}}^* + \boldsymbol{\omega} \times \boldsymbol{r}. \tag{5.32}$$

The acceleration a is defined as

$$\begin{aligned} \boldsymbol{a} &= \frac{d}{dt}[\boldsymbol{v}] && (5.33)\\ &= \frac{d}{dt}[\dot{\boldsymbol{r}}^* + \boldsymbol{\omega} \times \boldsymbol{r}] && (5.34)\\ &= (\ddot{\boldsymbol{r}}^{**} + \boldsymbol{\omega} \times \dot{\boldsymbol{r}}^*) + \dot{\boldsymbol{\omega}} \times \boldsymbol{r} + \boldsymbol{\omega} \times \dot{\boldsymbol{r}} && (5.35)\\ &= \ddot{\boldsymbol{r}}^* + \dot{\boldsymbol{\omega}} \times \boldsymbol{r} + 2\boldsymbol{\omega} \times \dot{\boldsymbol{r}}^* + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{r}). && (5.36) \end{aligned}$$

When the angular velocity $\boldsymbol{\omega}$ changes and an external angular velocity $\boldsymbol{\omega}_0$ acts upon the object, the angular acceleration is given as

$$\dot{\boldsymbol{\omega}} = \dot{\boldsymbol{\omega}}^* + \boldsymbol{\omega}_0 \times \boldsymbol{\omega}. \tag{5.37}$$

©Sethu Vijayakumar 1998

For the robot arm, the relative coordinates (or the body based coordinates) use the center of mass G as the point of origin and define coordinates along the major axes of the body system. The absolute coordinates (or the workspace coordinates) also use the center of mass G as the point of origin and define coordinates along the standard x, y and z coordinate system.

The moment of inertia for the absolute coordinates, $\hat{I}_x$,$\hat{I}_y$,$\hat{I}_z$ are defined as

$$\begin{cases} \hat{I}_x = \int (y^2 + z^2)\, dm \\ \hat{I}_y = \int (y^2 + x^2)\, dm \\ \hat{I}_z = \int (x^2 + y^2)\, dm \end{cases} \tag{5.38}$$

Here, $\hat{}$ refers to the values in absolute coordinates. Also, moments $\hat{I}_{xy}$,$\hat{I}_{yz}$,$\hat{I}_{zx}$ are defined as

$$\begin{cases} \hat{I}_{xy}(= \hat{I}_{yx}) = \int xy\, dm \\ \hat{I}_{yz}(= \hat{I}_{zy}) = \int yz\, dm \\ \hat{I}_{zx}(= \hat{I}_{xz}) = \int zx\, dm \end{cases} \tag{5.39}$$

Tensor matrix $\hat{\boldsymbol{I}}$ is given as

$$\hat{\boldsymbol{I}} = \begin{bmatrix} \hat{I}_x & -\hat{I}_{xy} & -\hat{I}_{xz} \\ -\hat{I}_{yx} & \hat{I}_y & -\hat{I}_{yz} \\ -\hat{I}_{zx} & -\hat{I}_{zy} & \hat{I}_z \end{bmatrix}. \tag{5.40}$$

This quantity corresponds to the mass for the angular motion. We can derive the tensor matrix corresponding to the relative coordinate system by using a rotation matrix R which gives the rotation through the center of the mass as

$$\boldsymbol{I} = R\hat{\boldsymbol{I}}R^t. \tag{5.41}$$

The angular momentum of a body performing angular motion $\boldsymbol{L}$ is given as

$$\boldsymbol{L} = \int_B (\boldsymbol{r} \times \dot{\boldsymbol{r}})\, dm. \tag{5.42}$$

Here, $\boldsymbol{r}$ corresponds to the position vector of every tiny mass (Δm) with the origin as the center of mass G, while $\int_B$ refers to the integration over the entire body B. Using the inertial tensor $\boldsymbol{I}$ and the angular velocity $\boldsymbol{\omega}$, $\boldsymbol{L}$ can be computed as

$$\boldsymbol{L} = \boldsymbol{I}\boldsymbol{\omega}. \tag{5.43}$$

When a body with center of mass G possessing an angular momentum of $\boldsymbol{L}$ rotates with an angular velocity of $\boldsymbol{\omega}$, the angular moment $\boldsymbol{N}$, which is the differential of $\boldsymbol{L}$ with respect to time is represented based on eq.(5.31) as

$$\boldsymbol{N} = \dot{\boldsymbol{L}}^* + \boldsymbol{\omega} \times \boldsymbol{L} \tag{5.44}$$

$$= \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} \tag{5.45}$$

$$= \boldsymbol{I}(\dot{\boldsymbol{\omega}}^* + \boldsymbol{\omega}_0 \times \boldsymbol{\omega}) + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}. \tag{5.46}$$

©Sethu Vijayakumar 1998

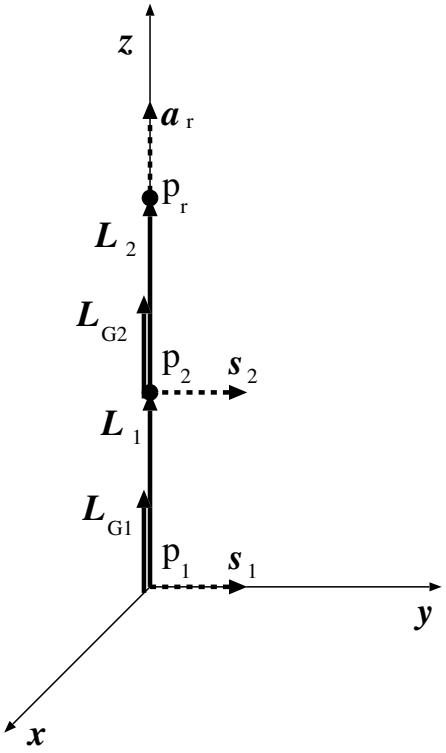


Figure 5.5: Idealized representation of 2DOF robot arm

A5.2 Deriving the inverse dynamics of the 2DOF robot arm

Here we will use the Newton-Euler method of estimation of the inverse dynamics. The basic steps involved are as follows:

1. Convert the motion equations in the relative(body) coordinates $(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})$ to the absolute(workspace) coordinate system.

2. Find the equations of motion with respect to the center of mass G_i of each link.

3. Derive the torques required at each link (starting from the outermost link moving towards the base) in order to achieve the required motion of the end effector.

The idealized figure of the 2DOF robot arm is shown in Fig.5.5.

Based on this model,

$$\hat{\boldsymbol{L}}_1 = [0 \;\; 0 \;\; L_1]^t, \tag{5.47}$$

©Sethu Vijayakumar 1998

$$\hat{\boldsymbol{L}}_2 = [0 \;\; 0 \;\; L_2]^t, \tag{5.48}$$

$$\hat{\boldsymbol{L}}_{G1} = [0 \;\; 0 \;\; L_{G1}]^t, \tag{5.49}$$

$$\hat{\boldsymbol{L}}_{G2} = [0 \;\; 0 \;\; L_{G2}]^t, \tag{5.50}$$

$$\boldsymbol{s}_1 = \boldsymbol{s}_2 = [0 \;\; 1 \;\; 0]^t. \tag{5.51}$$

The tensor matrices in the relative(body) coordinates are

$$\hat{\boldsymbol{I}}_1 = \begin{bmatrix} I_{1xy} & 0 & 0 \\ 0 & I_{1xy} & 0 \\ 0 & 0 & I_{1z} \end{bmatrix} \qquad \hat{\boldsymbol{I}}_2 = \begin{bmatrix} I_{2xy} & 0 & 0 \\ 0 & I_{2xy} & 0 \\ 0 & 0 & I_{2z} \end{bmatrix} \tag{5.52}$$

The rotation matrices are given as

$$R_1^0 = E^{j(\theta_1)} = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 \\ 0 & 1 & 0 \\ -\sin\theta_1 & 0 & \cos\theta_1 \end{bmatrix} \tag{5.53}$$

$$R_2^0 = E^{j(\theta_1+\theta_2)} = \begin{bmatrix} \cos(\theta_1+\theta_2) & 0 & \sin(\theta_1+\theta_2) \\ 0 & 1 & 0 \\ -\sin(\theta_1+\theta_2) & 0 & \cos(\theta_1+\theta_2) \end{bmatrix}. \tag{5.54}$$

Based on these, the vectors $\boldsymbol{L}_i$, $\boldsymbol{L}_{Gi}$ and the inertial tensor matrix $\boldsymbol{I}_i$ for link i in the absolute(workspace) coordinates can be written as

$$\boldsymbol{L}_1 = R_0^1 \hat{\boldsymbol{L}}_1 = L_2 \begin{bmatrix} \sin\theta_1 \\ 0 \\ \cos\theta_1 \end{bmatrix} \qquad \boldsymbol{L}_{G1} = R_0^1 \hat{\boldsymbol{L}}_{G1} = L_{G1} \begin{bmatrix} \sin\theta_1 \\ 0 \\ \cos\theta_1 \end{bmatrix} \tag{5.55}$$

$$\boldsymbol{L}_2 = R_0^2 \hat{\boldsymbol{L}}_2 = L_2 \begin{bmatrix} \sin\theta_1 \\ 0 \\ \cos\theta_1 \end{bmatrix} \qquad \boldsymbol{L}_{G2} = R_0^2 \hat{\boldsymbol{L}}_{G2} = L_{G2} \begin{bmatrix} \sin\theta_1 \\ 0 \\ \cos\theta_1 \end{bmatrix} \tag{5.56}$$

$$\boldsymbol{I}_1 = R_1^0 \hat{\boldsymbol{I}}_1 R_0^1 \tag{5.57}$$

$$= \begin{bmatrix} I_{1xy}\cos^2\theta_1 + I_{1z}\sin^2\theta_1 & 0 & (I_{1z} - I_{1xy})\sin\theta_1\cos\theta_1 \\ 0 & I_{1xy} & 0 \\ (I_{1z} - I_{1xy})\sin\theta_1\cos\theta_1 & 0 & I_{1xy}\sin^2\theta_1 + I_{1z}\cos^2\theta_1 \end{bmatrix} \tag{5.58}$$

$$\boldsymbol{I}_2 = R_2^0 \hat{\boldsymbol{I}}_2 R_0^2 \tag{5.59}$$

$$= \begin{bmatrix} I_{1xy}\cos^2(\theta_1+\theta_2) + I_{1z}\sin^2(\theta_1+\theta_2) & 0 & (I_{1z}-I_{1xy})\sin(\theta_1+\theta_2)\cos(\theta_1+\theta_2) \\ 0 & I_{1xy} & 0 \\ (I_{1z}-I_{1xy})\sin(\theta_1+\theta_2)\cos(\theta_1+\theta_2) & 0 & I_{1xy}\sin^2(\theta_1+\theta_2) + I_{1z}\cos^2(\theta_1+\theta_2) \end{bmatrix}$$

Based on the definitions above, we can compute the inverse dynamics by following the steps given below.

©Sethu Vijayakumar 1998

1. Derive the angular velocity vector $\boldsymbol{\omega}_i$:

First, we derive the angular velocity vector from the angular velocities of each link.

$$\boldsymbol{\omega}_1 = \boldsymbol{s}_1\dot{\theta_1} = \dot{\theta}_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{5.60}$$

$$\boldsymbol{\omega}_2 = \boldsymbol{s}_1\dot{\theta_1} + \boldsymbol{s}_2\dot{\theta_2} = (\dot{\theta}_1 + \dot{\theta}_2) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{5.61}$$

2. Derive the angular acceleration $\dot{\boldsymbol{\omega}}_i$:

Then, we use the angular velocity vector and the angular acceleration of each link to compute the angular acceleration vector.

$$\dot{\boldsymbol{\omega}}_1 = \boldsymbol{s}_1\ddot{\theta_1} = \ddot{\theta}_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{5.62}$$

$$\dot{\boldsymbol{\omega}}_2 = \boldsymbol{s}_1\ddot{\theta_1} + \boldsymbol{s}_2\ddot{\theta_2} + \boldsymbol{s}_1\dot{\theta_1} \times \boldsymbol{s}_2\dot{\theta_2} \tag{5.63}$$

$$= (\ddot{\theta}_1 + \ddot{\theta}_2) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{5.64}$$

3. Compute the acceleration $\boldsymbol{a}_i$:

We compute the acceleration vector $\boldsymbol{a}_i$ from the angular acceleration and the angular velocity vector.

$$\boldsymbol{a}_1 = 0 \tag{5.65}$$

$$\boldsymbol{a}_2 = \dot{\boldsymbol{\omega}}_1 \times \boldsymbol{L}_1 + \boldsymbol{\omega}_1 \times (\boldsymbol{\omega}_1 \times \boldsymbol{L}) \tag{5.66}$$

$$= \ddot{\theta}_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times L_1 \begin{bmatrix} \sin\theta_1 \\ 0 \\ \cos\theta_1 \end{bmatrix} + \dot{\theta}_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \left(\dot{\theta}_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} \sin\theta_1 \\ 0 \\ \cos\theta_1 \end{bmatrix} \right)$$

$$= L_1 \begin{bmatrix} \ddot{\theta}_1\cos\theta_1 - \dot{\theta}_1^2\sin\theta_1 \\ 0 \\ -\ddot{\theta}_1\sin\theta_1 - \dot{\theta}_1^2\cos\theta_1 \end{bmatrix} \tag{5.67}$$

$$\boldsymbol{a}_3 = \boldsymbol{a}_2 + \dot{\boldsymbol{\omega}}_2 \times \boldsymbol{L}_2 + \boldsymbol{\omega}_2 \times (\boldsymbol{\omega}_2 \times \boldsymbol{L}_2) \tag{5.68}$$

$$= \boldsymbol{a}_2 + (\ddot{\theta}_1 + \ddot{\theta}_2) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \text{Ł}_2 \begin{bmatrix} \sin(\theta_1 + \theta_2) \\ 0 \\ -\cos(\theta_1 + \theta_2) \end{bmatrix}$$

$$+ (\dot{\theta}_1 + \dot{\theta}_2) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \left((\dot{\theta}_1 + \dot{\theta}_2) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \text{Ł}_2 \begin{bmatrix} \sin(\theta_1 + \theta_2) \\ 0 \\ -\cos(\theta_1 + \theta_2) \end{bmatrix} \right)$$

©Sethu Vijayakumar 1998

$$= L_1 \begin{bmatrix} \ddot{\theta}_1 \cos\theta_1 - \dot{\theta}_1^2 \sin\theta_1 \\ 0 \\ -\ddot{\theta}_1 \sin\theta_1 - \dot{\theta}_1^2 \cos\theta_1 \end{bmatrix} \tag{5.69}$$

$$+ L_2 \begin{bmatrix} (\ddot{\theta}_1 + \ddot{\theta}_2)\cos(\theta_1+\theta_2) - (\dot{\theta}_1 + \dot{\theta}_2)^2 \sin(\theta_1+\theta_2) \\ 0 \\ -(\ddot{\theta}_1 + \ddot{\theta}_2)\sin(\theta_1+\theta_2) - (\dot{\theta}_1 + \dot{\theta}_2)^2 \cos(\theta_1+\theta_2) \end{bmatrix} \tag{5.70}$$

4. Compute the acceleration vector $\boldsymbol{a}_{Gi}$ of the center of mass G_i:

$$\boldsymbol{a}_{G1} = \boldsymbol{a}_1 + \dot{\boldsymbol{\omega}}_1 \times \boldsymbol{L}_{G1} + \boldsymbol{\omega}_1 \times (\boldsymbol{\omega}_1 \times \boldsymbol{L}_{G1}) \tag{5.71}$$

$$= \boldsymbol{a}_1 + \ddot{\theta}_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times L_{G1} \begin{bmatrix} \sin\theta_1 \\ 0 \\ \cos\theta_1 \end{bmatrix} \tag{5.72}$$

$$+ \dot{\theta}_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \left(\dot{\theta}_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} \sin\theta_1 \\ 0 \\ \cos\theta_1 \end{bmatrix} \right) \tag{5.73}$$

$$= L_{G1} \begin{bmatrix} \ddot{\theta}_1 \cos\theta_1 - \dot{\theta}_1^2 \sin\theta_1 \\ 0 \\ -\ddot{\theta}_1 \sin\theta_1 - \dot{\theta}_1^2 \cos\theta_1 \end{bmatrix} \tag{5.74}$$

$$\boldsymbol{a}_{G2} = \boldsymbol{a}_2 + \dot{\boldsymbol{\omega}}_2 \times \boldsymbol{L}_{G2} + \boldsymbol{\omega}_2 \times (\boldsymbol{\omega}_2 \times \boldsymbol{L}_{G2}) \tag{5.75}$$

$$= L_1 \begin{bmatrix} \ddot{\theta}_1 \cos\theta_1 - \dot{\theta}_1^2 \sin\theta_1 \\ 0 \\ -\ddot{\theta}_1 \sin\theta_1 - \dot{\theta}_1^2 \cos\theta_1 \end{bmatrix} + L_{G2}(\ddot{\theta}_1 + \ddot{\theta}_2) \begin{bmatrix} \cos(\theta_1+\theta_2) \\ 0 \\ -\sin(\theta_1+\theta_2) \end{bmatrix}$$

$$+ L_{G2}(\dot{\theta}_1 + \dot{\theta}_2)^2 \begin{bmatrix} -\sin(\theta_1+\theta_2) \\ 0 \\ -\cos(\theta_1+\theta_2) \end{bmatrix}$$

$$= L_1 \begin{bmatrix} \ddot{\theta}_1 \cos\theta_1 - \dot{\theta}_1^2 \sin\theta_1 \\ 0 \\ -\ddot{\theta}_1 \sin\theta_1 - \dot{\theta}_1^2 \cos\theta_1 \end{bmatrix}$$

$$+ L_{G2} \begin{bmatrix} (\ddot{\theta}_1 + \ddot{\theta}_2)\cos(\theta_1+\theta_2) - (\dot{\theta}_1 + \dot{\theta}_2)^2 \sin(\theta_1+\theta_2) \\ 0 \\ -(\ddot{\theta}_1 + \ddot{\theta}_2)\sin(\theta_1+\theta_2) - (\dot{\theta}_1 + \dot{\theta}_2)^2 \cos(\theta_1+\theta_2) \end{bmatrix} \tag{5.76}$$

5. Compute the force acting on the center of mass $\boldsymbol{F}_{Gi}$:

$$\boldsymbol{F}_{G1} = m_1(\boldsymbol{a}_{G1} - \boldsymbol{g}) = m_1 L_{G1} \begin{bmatrix} \ddot{\theta}_1 \cos\theta_1 - \dot{\theta}1^2 \sin\theta_1 \\ 0 \\ g/L_{G1} - \ddot{\theta}_1 \sin\theta_1 - \dot{\theta}_1^2 \cos\theta_1 \end{bmatrix} \tag{5.77}$$

©Sethu Vijayakumar 1998

$$\begin{aligned}
\boldsymbol{F}_{G2} &= m_2(\boldsymbol{a}_{G2} - \boldsymbol{g}) && (5.78)\\
&= m_2 L_1 \begin{bmatrix} \ddot{\theta}_1 \cos\theta_1 - \dot{\theta}_1^2 \sin\theta_1 \\ 0 \\ g/L_1 - \ddot{\theta}_1 \sin\theta_1 - \dot{\theta}_1^2 \cos\theta_1 \end{bmatrix} \\
&\quad + m_2 L_{G2} \begin{bmatrix} (\ddot{\theta}_1 + \ddot{\theta}_2)\cos(\theta_1+\theta_2) - (\dot{\theta}_1 + \dot{\theta}_2)^2 \sin(\theta_1+\theta_2) \\ 0 \\ -(\ddot{\theta}_1 + \ddot{\theta}_2)\sin(\theta_1+\theta_2) - (\dot{\theta}_1 + \dot{\theta}_2)^2 \cos(\theta_1+\theta_2) \end{bmatrix}
\end{aligned}$$

6. Compute the moment $\boldsymbol{N}_{Gi}$ acting on the center of mass G_i:

$$\begin{aligned}
\boldsymbol{N}_{G1} &= \boldsymbol{I}_1 \dot{\boldsymbol{\omega}}_1 + \boldsymbol{\omega}_1 \times (\boldsymbol{I}_1 \boldsymbol{\omega}_1) && (5.79)\\
&= \boldsymbol{I}_1 \ddot{\theta}_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \dot{\theta}_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \left(\boldsymbol{I}_1 \dot{\theta}_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right) && (5.80)\\
&= I_{1xy} \ddot{\theta}_1 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} && (5.81)\\
\boldsymbol{N}_{G2} &= \boldsymbol{I}_2 \dot{\boldsymbol{\omega}}_2 + \boldsymbol{\omega}_2 \times (\boldsymbol{I}_2 \boldsymbol{\omega}_2) && (5.82)\\
&= \boldsymbol{I}_2 (\ddot{\theta}_1 + \ddot{\theta}_2) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + (\dot{\theta}_1 + \dot{\theta}_2) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \left(\boldsymbol{I}_2 (\dot{\theta}_1 + \dot{\theta}_2) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right) \\
&= I_{2xy} (\ddot{\theta}_1 + \ddot{\theta}_2) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} && (5.83)\\
& && (5.84)
\end{aligned}$$

7. Compute the force F_i acting on each link:

$$\begin{aligned}
\boldsymbol{F}_2 &= \boldsymbol{F}_{G2} && (5.85)\\
\boldsymbol{F}_1 &= \boldsymbol{F}_{G1} + \boldsymbol{F}_{G2} && (5.86)
\end{aligned}$$

8. Compute the moment $\boldsymbol{N}_i$ acting on each link:

$$\begin{aligned}
\boldsymbol{N}_2 &= \boldsymbol{L}_{G2} \times \boldsymbol{F}_{G2} + \boldsymbol{N}_{g2} && (5.87)\\
&= L_{G2} \begin{bmatrix} \sin(\theta_1+\theta_2) \\ 0 \\ \cos(\theta_1+\theta_2) \end{bmatrix} \times m_2 \begin{bmatrix} \dots \\ 0 \\ \dots \end{bmatrix} + I_{2xy} (\ddot{\theta}_1 + \ddot{\theta}_2) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} && (5.88)
\end{aligned}$$

©Sethu Vijayakumar 1998

$$= \begin{bmatrix} 0 \\ N_{2y} \\ 0 \end{bmatrix} \tag{5.89}$$

$$\boldsymbol{N}_1 = \boldsymbol{N}_2 + \boldsymbol{L}_1 \times \boldsymbol{F}_2 + \boldsymbol{L}_{G1} \times \boldsymbol{F}_{G1} + \boldsymbol{N}_{G1} \tag{5.90}$$

$$= \begin{bmatrix} 0 \\ N_{1y} \\ 0 \end{bmatrix} \tag{5.91}$$

Here, N_{iy} is computed as

$$\begin{aligned} N_{2y} &= (m_2 L_{G2}^2 + m_2 L_1 L_{G2} \cos\theta_2 + I_{2xy})\,\ddot{\theta}_1 + (m_2 L_{G2}^2 + I_{2xy})\,\ddot{\theta}_2 \\ &\quad + m_2 L_1 L_{G2} \dot{\theta}_1^2 \sin\theta_2 - m_2 L_{G2} g \sin(\theta_1 + \theta_2) \end{aligned} \tag{5.92}$$

$$\begin{aligned} N_{1y} &= (I_{1xy} + I_{2xy} + m_1 {L_{G1}}^2 + m_2 {L_{G2}}^2 + m_2 L_1^2 + 2 m_2 L_1 L_{G2} \cos\theta_2)\,\ddot{\theta}_1 \\ &\quad + (I_{2xy} + m_2 {L_{G2}}^2 + m_2 L_1 L_{G2} \cos\theta_2)\,\ddot{\theta}_2 \\ &\quad - m_1 L_1 L_{G2} \sin\theta_2 (\dot{\theta}_2^2 + \dot{\theta}_1 \dot{\theta}_2) \\ &\quad - \Big\{ m_1 L_{G1} \sin\theta_1 + m_2 \Big(L_1 \sin\theta_1 + \mathrm{L}_{G2} \sin(\theta_1 + \theta_2) \Big) \Big\}\, g \end{aligned} \tag{5.93}$$

9. Torques τ_i acting on each link:

We compute the torques τ_i that has to be applied to each link based on the moment computed earlier.

$$\tau_1 = \boldsymbol{s}_1^t \boldsymbol{N}_1 \tag{5.94}$$

$$= [\,0 \;\; 1 \;\; 0\,]\,\boldsymbol{N}_1 = N_{1y} \tag{5.95}$$

$$\tau_2 = \boldsymbol{s}_2^t \boldsymbol{N}_2 \tag{5.96}$$

$$= [\,0 \;\; 1 \;\; 0\,]\,\boldsymbol{N}_2 = N_{2y} \tag{5.97}$$

©Sethu Vijayakumar 1998

©Sethu Vijayakumar 1998

6

Active Data Selection for Optimal Generalization

6.1 The active learning framework

We have discussed the problem of learning by considering it as a non-linear function approximation problem. Effective methods of optimizing the learned function, both under batch (Chapter 2) and incremental (Chapter 4) conditions corresponding to various optimization criteria has been derived. However, in all these approaches, we are doing 'learning from examples' in a *passive* manner, i.e., we use the given training data and have no control on the location or type of training examples.

It has been well known (and has been illustrated through a toy example in Section 4.10.2) that supervised learning is highly data dependent. The level of generalization ability achievable with a fixed number of training set (non-asymptotic analysis) is heavily dependent on the quality of the data used. It is also interesting to note that many natural learning systems are not simply passive, but make use of at least some form of active learning to examine the problem domain. By *active* learning, we mean any form of learning in which the learning program has some control over the inputs over which it trains. In natural systems (such as humans), this phenomenon is exhibited at both high levels (e.g. active examination of objects) and low, subconscious level (e.g. Fernald and Kulh's [FK87] work on infant reactions to 'Motherese' speech).

Typically, the improvement in generalization performance by active learning is obtained at an increased computational expense [Coh94]. However, the purpose of such

increased cost is to concentrate gathered information in the more useful regions of the search space. A smaller training data set gathered by an active learner produces generalization performance equal to or better than a much larger data set containing redundant examples [SS95]. It follows that active learning methods are best recommended in situations where the cost of obtaining training samples far outweighs the additional computational cost of an automated data gathering strategy.

6.2 Various information measure and approaches

The problem of "active learning" has been extensively studied in economic theory and statistics [Fed72]. Optimal data selection within the Bayesian framework for interpolation have been studied by Luttrell[Lut85] and MacKay[Mac92]. Plutowski and White [PW93] study a different problem in the context of noise free interpolation: they assume that a large amount of data has been collected and work on principles of selecting a subset of that data for efficient training; the entire data sets (inputs and outputs) is consulted at each iteration to decide which example to add, an option that is not permitted in this work.

When the problem of deciding which training data to select next comes up, the question of comparing the available data options obviously arise. Our criterion for how informative a new datum is will depend on what we are interested in.

Several alternatives that can be considered are :

1. We might want the new data to be maximally informative in a global sense, i.e., reduce the overall error term to the maximum.

2. Alternatively, we might not be interested in getting a globally informative next data; instead, we might want to get a better approximation around the region where we may not be able to sample directly.

3. Lastly, we might want data that maximally helps in distinguishing between the possible options; for example, in a probabilistic model, one which helps in eliminating the possibility of some model while confirming the other.

The aim would be to find a utility as a function of the next sample location x_{m+1}. This utility is often referred to as the *information measure* of the new data. As for the evaluation of the information measure is concerned, a lot of work has been done in different fields [Lut85, Mac92, WHG90]. Two measures of information has been suggested, both based on Shannon's entropy, whose properties as a sensible information measure are well known. Both these approaches, being Bayesian in

©Sethu Vijayakumar 1998

their core, assume a probability distribution for the parameters of the approximating model, w. Let the probability distribution of the parameters before and after we receive the new datum (x_{m+1}, y_{m+1}) be $P^m(w)$ and $P^{m+1}(w)$, respectively. Then, the *change in entropy* (refer [DO96]) of the distribution is $\Delta S = S_m - S_{m+1}$, where

$$S_m = \int P^m(w) log \frac{m(w)}{P^m(w)} dw, \tag{6.1}$$

and m is a measure on w that is inconsequential and can be taken as a constant. The greater ΔS is, the more information we have gained about w.

An alternate information measure is the *cross entropy* between $P^m(w)$ and $P^{m+1}(w)$:

$$G = -\int P^{m+1}(w) log \frac{P^m(w)}{P^{m+1}(w)} dw. \tag{6.2}$$

G is a measure of how much information we gain when we are informed that the true distribution of w is $P^{m+1}(w)$ rather than $P^m(w)$.

Apart from this there are various other approaches including using the Fisher Information criterion [Fuk96]. Here, we will assume that we are provided with the prior distribution of the possible functions over the search space, in other words, we have the correlation matrix Q defined in eq.(2.18). Then, it is appropriate to use the Wiener optimization criterion to serve as a measure of the information gained due to the newly sampled data, since the *reduction of error* over the new sample points can be viewed as *new information gained*. Therefore, this can be interpreted as looking for globally informative data which reduces the error in an averaged sense over the original function space H. In the next section, we shall formalize the Wiener criterion based on incremental sampling in order to find a framework for selecting optimal training data.

6.3 The Wiener criterion based information measure

In this section, we shall use the Wiener criterion described in Section 2.3.3 as an information measure for deciding the usefulness of the training samples. We can look at different cases of active learning such as starting with a set of m training data and selecting the next data optimally (constrained optimal solution), selecting the set of $m+1$ training data in one *batch*, not using the m data selected so far but instead select a completely new optimal $m+1$ data set or even (if feasible) select only the $m+1$st data point such that the entire set $\{x_n : 1 \leq n \leq m+1\}$ is optimal. The various possibilities that can be considered are summarized in Table 6.1. Essentially, for all the above listed cases, we have to consider two different types of optimization; one in which the variable parameters to be optimized range over

©Sethu Vijayakumar 1998

Table 6.1: Types of active learning

Do not use given data	**Use given data**	
	Optimal solution	Constrained optimal solution
Find optimal $\{x_n\}_{n=1}^{m+1}$.	Find x_{m+1} such that the entire set of $m+1$ data points are optimal.	Find x_{m+1} such that the $m+1$ st data point is optimal, given m data points.
(Section 6.3.1)	(Section 6.3.2)	(Section 6.3.2)

the entire training samples and hence, the entire sampling operator and the other, in which the optimization can be done only based on the next training data as the variable. Optimization criterion for both these cases will be considered next.

6.3.1 Criterion for batch selection of complete data set

The criterion to be minimized for active selection of the training set, based on the active learning formulation of eq.(2.17), is given below.

$$\min_{X_{m+1}, A_{m+1}} J_W[X_{m+1}, A_{m+1}] \tag{6.3}$$

where

$$J_W[X_{m+1}, A_{m+1}] \equiv E_f \|X_{m+1} A_{m+1} f - f\|^2 \tag{6.4}$$

As can be seen from eq.(6.3), actually we ought to minimize the functional by changing both the sampling operator A and the learned operator X simultaneously. However, here we use a substitute or modified criteria due to ease of solution. The modified criterion is as given below.

$$\min_{A_{m+1}} J_W^{(0)}[A_{m+1}] \tag{6.5}$$

where

$$J_W^{(0)}[A_{m+1}] \equiv \min_{X_{m+1}} J_W[X_{m+1}] \tag{6.6}$$

©Sethu Vijayakumar 1998

$$J_W[X_{m+1}] \equiv E_f \|X_{m+1}A_{m+1}f - f\|^2 \tag{6.7}$$

We will use this criterion to solve for optimal data selection in the next section.

6.3.2 Criterion for constrained optimal selection of next data

Here, we consider the case of going from a stage of m sample points to $m+1$ sample points, i.e., in other words, doing an optimization under the constraint of using the so far provided m training samples. Eq.(2.17) representing the Wiener optimization can be formulated in view of incremental *active learning* theory as:

$$\text{minimize over } x_{m+1} : J^{(0)}_{Wm+1}[x_{m+1}], \tag{6.8}$$

where

$$J^{(0)}_{Wm+1}[x_{m+1}] \equiv \min_{X_{m+1}} J_{Wm+1}[X_{m+1}], \tag{6.9}$$

$$J_{Wm+1}[X_{m+1}] \equiv E_f \|X_{m+1}A_{m+1}f - f\|^2. \tag{6.10}$$

The modified optimization criterion now involves a two level optimization. This involves finding the most optimal next sample point x_{m+1} which minimizes the error term $J^{(0)}_{Wm+1}[x_{m+1}]$; where $J^{(0)}_{Wm+1}$ is the optimization on the learning operator X_{m+1} for a given training set including the new data point x_{m+1}. This is also a substitute for a true criterion which actually involves changing both the learning operator X and the location of the new sample point *simultaneously*.

6.4 Data selection strategy

In this section, we will evaluate a strategy for batch selection of the training data such that the optimization criterion described in Section 6.3.1 is minimized.

We already know the solution for the optimal learning operator X_{m+1} to satisfy the Wiener criterion under fixed or given training data (solution to the functional of eq.(6.6)). The necessary and sufficient condition for the criterion to be satisfied is given as

$$X_{m+1}A_{m+1}RA^*_{m+1} = RA^*_{m+1} \tag{6.11}$$

Solving this, we have the general form of the solution given as

$$X_{m+1} = RA^*_{m+1}(A_{m+1}RA^*_{m+1})^\dagger + W(I - UU^\dagger) \tag{6.12}$$

©Sethu Vijayakumar 1998

where $R = E_f(f \otimes \overline{f})$, $U = (A_{m+1} R A_{m+1}^*)$ and W is an arbitrary operator.

Now, proceeding with the optimization criterion and evaluating the functional of eq.(6.7), we have

$$\begin{aligned} J_W[X_{m+1}] &= \underset{f}{E} \|X_{m+1}A_{m+1}f - f\|^2 \\ &= \underset{f}{E}\, tr\{(X_{m+1}A_{m+1} - I)(f \otimes \overline{f})(X_{m+1}A_{m+1} - I)^*\} \\ &= tr\{(X_{m+1}A_{m+1} - I)R(X_{m+1}A_{m+1} - I)^*\} \\ &= tr(X_{m+1}A_{m+1}RA_{m+1}^*X_{m+1}^* - X_{m+1}A_{m+1}R \\ &\quad - RA_{m+1}^*X_{m+1}^* + R) \end{aligned} \tag{6.13}$$

Using the result of eq.(6.12) and eq.(6.13) in the functional of eq.(6.5), we have

$$\begin{aligned} \min_{A_{m+1}} J_W^{(0)}[A_{m+1}] &= \min_{A_{m+1}} tr(RA_{m+1}^*X_{m+1}^* - X_{m+1}A_{m+1}R \\ &\qquad - RA_{m+1}^*X_{m+1}^* + R) \\ &= \min_{A_{m+1}} \{tr(R) - tr(X_{m+1}A_{m+1}R)\} \end{aligned} \tag{6.14}$$

Therefore, we can convert the problem of minimizing the functional described in eq.(6.5) to the following equivalent maximization problem.

$$\min_{A_{m+1}} J_W^{(0)}[A_{m+1}] \equiv \max_{A_{m+1}} J_W'[A_{m+1}] \tag{6.15}$$

where

$$J_W'[A_{m+1}] = tr(X_{m+1}A_{m+1}R), \tag{6.16}$$

since $tr(R)$ is a quantity that gets fixed based on the apriori knowledge about the function space of the problem under consideration.

Using the eq.(6.12) with $W = 0$ in the functional of eq.(6.15), we have

$$\begin{aligned} J_W'[A_{m+1}] &= tr(X_{m+1}A_{m+1}R) \\ &= tr\{RA_{m+1}^*(A_{m+1}RA_{m+1}^*)^\dagger A_{m+1}R\} \\ &= (RA_{m+1}^*(A_{m+1}RA_{m+1}^*)^\dagger A_{m+1}R, I) \\ &= ((A_{m+1}RA_{m+1}^*)^\dagger A_{m+1}R, A_{m+1}R) \end{aligned} \tag{6.17}$$

Let us substitute $T = A_{m+1}R^{\frac{1}{2}}$ in the above functional. Then, we have

$$\begin{aligned} J_W'[A_{m+1}] &= ((TT^*)^\dagger TR^{\frac{1}{2}}, TR^{\frac{1}{2}}) \\ &= (T^*(TT^*)^\dagger T, R) \\ &= (T^\dagger T, R) \\ J_W'[A_{m+1}] &= (P_{\mathcal{R}(T^*)}, R) \end{aligned} \tag{6.18}$$

©Sethu Vijayakumar 1998

where $P_{\mathcal{R}(T^*)}$ is an orthogonal projection operator onto the range of $\mathcal{R}(R^{\frac{1}{2}}A^*_{m+1})$. From now on, in the intermediate equations, we will represent this projection operator as P without specifying the subspace of projection. Now, the functional of eq.(6.18) becomes

$$J'_W[P] = (P, R) = (PR, P) \tag{6.19}$$

based on the property of orthogonal projection operators $P^* = P$ and $P^2 = P$. An operator P is an orthogonal projection operator if and only if

$$P^*P = P. \tag{6.20}$$

Furthermore, if we assume the dimensionality of the subspace $R^{\frac{1}{2}}A^*_{m+1}$ to be equal to L, then, we have

$$tr(P^*P) = L. \tag{6.21}$$

Hence, our problem can be restated as **"Obtain an operator P which maximizes eq.(6.19) under constraints eqs.(6.20) and (6.21)"**.

Let C and λ be the Lagrange multiplier operator and the Lagrange multiplier, respectively. Then, the above variational problem with constraints is reduced to the following variational problem without constraints with respect to P, C and λ

$$J'_W[P, C, \lambda] = (PR, P) + 2Re(C, P^*P - P) + \lambda[tr(P^*P) - L]. \tag{6.22}$$

Since we have to maximize this functional, equating the partial derivative of J in eq.(6.22) with respect to P to zero yields

$$Re\{(P(R + C + C^* + \lambda P) - C), \delta P\} = 0 \tag{6.23}$$

which in turn gives the relationship

$$P(R + C + C^* + \lambda P) = C \tag{6.24}$$

Using the properties of the projection operator P and the fact that $R^* = R$, we can prove the following lemma, as shown in Appendix A6.1.

Lemma 6.1 *The functional represented by eq.(6.22) is maximized only if*

$$PR = RP. \tag{6.25}$$

We can convert the result of the above lemma into more useful form through the following results. The proof of these, on the lines of analysis in [Oga92a], is fairly straightforward.

©Sethu Vijayakumar 1998

Lemma 6.2

$$PR = RP$$

if and only if

$$R\mathcal{R}(R^{\frac{1}{2}}A^*_{m+1}) \subseteq \mathcal{R}(R^{\frac{1}{2}}A^*_{m+1}), \tag{6.26}$$

where P is the orthogonal projection onto $S = \mathcal{R}(R^{\frac{1}{2}}A^*_{m+1})$

Lemma 6.3

$$R\mathcal{R}(R^{\frac{1}{2}}A^*_{m+1}) \subseteq \mathcal{R}(R^{\frac{1}{2}}A^*_{m+1})$$

if and only if $\mathcal{R}(R^{\frac{1}{2}}A^*_{m+1})$ is a Karhunen-Loéve subspace of the kernel R.

The above lemma means that when the relation $PR = RP$ holds, the subspace $\mathcal{R}(R^{\frac{1}{2}}A^*_{m+1})$ is spanned by the eigen vectors of R. Let λ_n be the n-th eigen value of the correlation operator R arranged in decreasing order and φ_n the corresponding eigen vector with norm = 1, i.e.

$$R\varphi_n = \lambda_n\varphi_n \quad (\lambda_1 \geq \lambda_2 \geq ...). \tag{6.27}$$

Then, P can be represented, due to Lemma 6.3, as

$$P = \sum_{n=1}^{L} (\varphi_{m_n} \otimes \overline{\varphi_{m_n}}). \tag{6.28}$$

where $\{m_n : 1 \leq n \leq L\}$ is a set of indices. Hence, our problem has been reduced to that of obtaining a set of eigen vectors $\{\varphi_{m_n} : 1 \leq n \leq L\}$ which maximizes eq.(6.19). Since the functional being maximized, $J'_W[P]$, is given as

$$J'_W[P] = (P, R), \tag{6.29}$$

from eq.(6.28), we have,

$$\begin{aligned} J'_W[P] &= (R, P) && (6.30)\\ &= \sum_{n=1}^{L} (R\varphi_{m_n}, \varphi_{m_n}) && (6.31)\\ &= \sum_{n=1}^{L} \lambda_{m_n} && (6.32) \end{aligned}$$

Since λ_n are arranged in decreasing order, eq.(6.19) is maximized if and only if we take

$$\{\lambda_{m_n} :: 1 \leq n \leq L\} = \{\lambda_n :: 1 \leq n \leq L\} \tag{6.33}$$

Hence, to maximize the functional and select the optimal set of training data, we should use a sampling operator A_{m+1} such that $\mathcal{R}(R^{\frac{1}{2}}A^*_{m+1})$ is the subspace spanned

©Sethu Vijayakumar 1998

by $\mathcal{L}\{\varphi_n\}_{n=1}^{L}$, where φ_n are the eigen vectors corresponding to the L largest eigen values λ_n of the correlation operator R.

Based on the above analysis, we can write a theorem summarizing the necessary and sufficient condition for selecting the optimal training set.

Theorem 6.1 (Optimal selection of training data)
The necessary and sufficient condition for the optimization of the functional for optimal training data selection, represented by eq.(6.19) is that $\mathcal{R}(R^{\frac{1}{2}}A_{m+1}^{*})$ is the subspace spanned by $\mathcal{L}\{\varphi_n\}_{n=1}^{L}$ where $L = dim(\mathcal{R}(R^{\frac{1}{2}}A_{m+1}^{*}))$ and φ_n are the eigen vectors corresponding to the L largest eigen values λ_n of the correlation operator R.

6.5 Comments

Here, we will further analyze the implications of the Theorem 6.1 on optimal training data selection. Since, the sampling operator A_{m+1} is defined as

$$A_{m+1} = \sum_{k=1}^{m+1} (e_k \otimes \overline{\psi_k}), \tag{6.34}$$

we know that

$$\mathcal{R}(A_{m+1}^{*}) = \mathcal{L}\{\psi_k\}_{k=1}^{m+1} \tag{6.35}$$

and

$$\mathcal{R}(R^{\frac{1}{2}}A_{m+1}^{*}) = \mathcal{L}\{R^{\frac{1}{2}}\psi_k\}_{k=1}^{m+1}. \tag{6.36}$$

Also based on the eigen value decomposition of R, we can express the correlation operator and it's root as

$$\begin{aligned} R &= \sum_{n=1}^{\infty} \lambda_n(\varphi_n \otimes \overline{\varphi_n}) \\ R^{\frac{1}{2}} &= \sum_{n=1}^{\infty} \sqrt{\lambda_n}(\varphi_n \otimes \overline{\varphi_n}) \end{aligned} \tag{6.37}$$

Hence, using the result of eq.(6.37), we can show that

$$\begin{aligned} R^{\frac{1}{2}}\psi_k(x) &= \sum_{n=1}^{\infty} \sqrt{\lambda_n}(\varphi_n \otimes \overline{\varphi_n})\psi_k(x) \\ &= \sum_{n=1}^{\infty} \sqrt{\lambda_n}(\varphi_n, \psi_k)\varphi_n(x) \\ &= \sum_{n=1}^{\infty} \sqrt{\lambda_n}\varphi_n(x_k)\varphi_n(x). \end{aligned} \tag{6.38}$$

©Sethu Vijayakumar 1998

Now, turning our attention to the condition to be satisfied for optimal training data selection, we know from Theorem 6.1 that the necessary and sufficient condition for optimal sampling is that the following relation holds:

$$R^{\frac{1}{2}}\mathcal{R}(A^*_{m+1}) = \mathcal{L}\{\varphi_n\}_{n=1}^{L}, \tag{6.39}$$

where $L = dim(\mathcal{R}(R^{\frac{1}{2}}A^*_{m+1}))$ and $\mathcal{L}\{\varphi_n\}_{n=1}^{L}$ is the maximum variance subspace of the correlation operator. Based on eq.(6.36), we can write the relation of eq.(6.39) as

$$\mathcal{L}\{R^{\frac{1}{2}}\psi_k\}_{k=1}^{m+1} = \mathcal{L}\{\varphi_n\}_{n=1}^{L}, \tag{6.40}$$

which, due to eqs.(6.38), is equivalent to

$$\mathcal{L}\{\sum_{n=1}^{\infty}\sqrt{\lambda_n}\varphi_n(x_k)\varphi_n\}_{k=1}^{m+1} = \mathcal{L}\{\varphi_n\}_{n=1}^{L}. \tag{6.41}$$

Whether the condition specified in eq.(6.41) can always be satisfied will depend on the nature of the correlation operator and the function space being used.

A more general solution can be obtained by changing the step where we substituted the minimization of the functional of eq.(6.18) with eq.(6.19). The point to be noted here is that we solved for a general projection operator P whereas in the original function, the projection operator was a projection onto a particular subspace $\mathcal{R}(R^{\frac{1}{2}}A^*_{m+1})$. This will be pursued as one of the future goals in line with this research. Moreover, training data selection in incremental fashion, i.e., optimally selecting $m+1$st data only under constraint of using the previous m data is a topic for future work.

©Sethu Vijayakumar 1998

Appendix A6

A6.1 Proof of Lemma 6.1

The maximization of the functional eq.(6.22) is equivalent to solving for P in the eq.(6.24),

$$P(R + C + C^* + \lambda P) = C. \tag{6.42}$$

Multiplying eq.(6.42) with P from the left, we get the relation

$$PR + PC^* + \lambda P = 0. \tag{6.43}$$

Multiplying the conjugate of eq.(6.42) with P from the right, we have

$$RP + CP + \lambda P = 0. \tag{6.44}$$

Multiplying eq.(6.42) with P from the right, we get the relation

$$P(R + C + C^* + \lambda P)P = CP, \tag{6.45}$$

which in turn implies (by taking conjugate and equating) that

$$CP = (CP)^* = PC^*. \tag{6.46}$$

Combining eqs.(6.43), (6.44), and (6.46), we have

$$PR = RP,$$

which completes the proof.

©Sethu Vijayakumar 1998

©Sethu Vijayakumar 1998

Part III

Incremental Learning in High Dimensional Spaces

7

Local Dimensionality Reduction

7.1 The curse of dimensionality

In Part I of this dissertation, we have laid out a statistical framework based on functional analysis theory and developed techniques for learning with optimal generalization with respect to a particular optimization criterion. Exact incremental solutions for these were provided along with an analysis of methods for selecting optimal training data (active learning). In the theory developed so far, selection of the right model space is extremely critical, as is the case with all global learning techniques. Guidelines to do this effectively based on nature of problem was provided in Chapter 3 and effects of these were illustrated through examples.

However, as the complexity of the problem; in terms of the original functions space or the number of input dimensions increase, it becomes more and more difficult to obtain the right model space or find the right bias. As shown in Chapter 5, when we look at learning the sensorimotor map of a 2DOF robot arm, we still can use the physical dynamics of the system to construct our search space. However, as the problem gets more complex (for example, in learning the sensorimotor map of a 7 DOF robot arm; which is a 21 to 7 dimensional mapping), it becomes very difficult to use parametric models due to lack of enough apriori information about the original function space.

Hence, we delve into the area of non-parametric regression using locally weighted linear learning models. Local linear models have a well established track record in

situations with enough data support [AMS97] and are known to effectively circumvent the problem of bias/variance dilemma [GBD92]. We will talk about these and their pros and cons in more detail in Chapter 8.

However, even local weighted learning, although managing to circumvent the direct model selection problem, suffer from the curse of dimensionality. For example, Receptive Field Locally Weighted Regression (RFWR) introduced in Schaal and Atkeson [SA96] is one of the local learning schemes that have been shown to possess favorable learning properties for incremental learning without interference. But, computational complexity of RFWR increases more than quadratic with the number of inputs to the system, a fact which confined the application of RFWR to problems of low dimensionality. In addition, nonparametric learning methods that depend on the notion of "neighborhood" generally scale unfavorably to high dimensions. The reason for this behavior comes from the non-intuitive effect that in high dimensional spaces, e.g., 20-dimensional, all data points are approximately the same distance away from each other [Sco92], thus destroying the discriminative power of neighborhood relations.

7.2 Evidence towards locally low dimensional structure in motion data

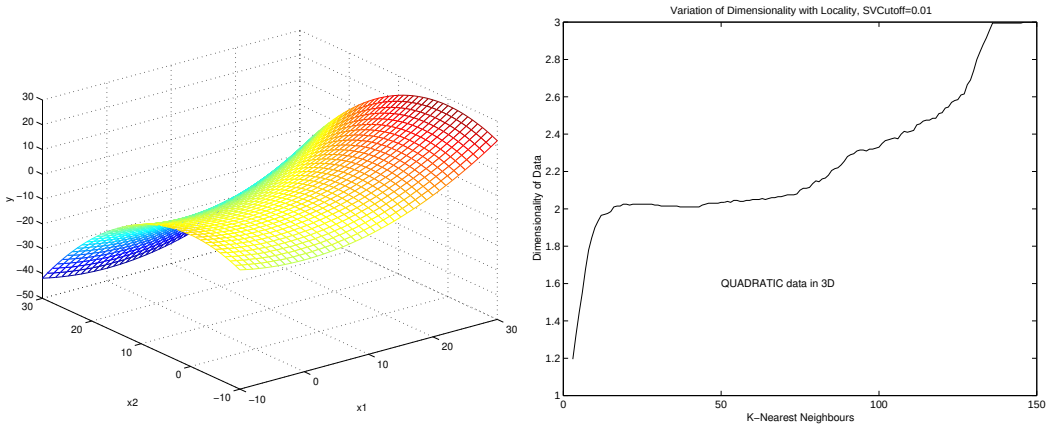
Given this "curse of dimensionality", nonparametric learning systems – and actually all other general nonlinear learning systems – seem to have limited merits for learning in high dimensional problems like that of sensorimotor control. However, when one examines data distributions of high dimensional data sets generated from real physical systems, one often notices that *locally* such data is not high dimensional at all and rarely exceed 5-8 dimensions. It is this observation which motivates the approach suggested in the second part of this dissertation.

7.2.1 The dimensionality analysis tool

To analyze the local dimensionality of certain data sets, we developed a tool which looked at the Singular Value Decomposition (SVD) of a data in various degrees of locality. Then, we used a certain threshold to compare the variance in the data by looking at what percentage of the variance is accounted for by projections of these data onto a lower subspace.

Formalizing this concept, let us assume we look at an N dimensional input data

©Sethu Vijayakumar 1998



(a) (b)

Figure 7.1: (a) A synthetic 3 dimensional data set consisting of a quadratic surface. (b)The variation in local dimensionality with decreasing locality for this data.

space. First of all, we decide on the region of locality and consider only data points falling within this region. We compute the singular values, $\{s_i\}_{i=1}^{N}$, of the covariance matrix of this local data, and sort such that $s_1 \geq s_2 \geq ... \geq s_N$. The dimensionality of a local region is defined as the number of singular values above a certain ratio threshold, where the threshold is the user defined SVD cutoff, θ_{SVD}, in the analysis. Mathematically, if we define L such that $L = N$ if

$$\frac{s_N}{\sum_{i=1}^{N} s_i} > \theta_{SVD} \tag{7.1}$$

else $L = K$, where K satisfies the relation

$$\frac{s_{K+1}}{\sum_{i=1}^{K+1} s_i} \leq \theta_{SVD} < \frac{s_K}{\sum_{i=1}^{K} s_i} : 1 \leq K < N, \tag{7.2}$$

then, L is the dimensionality of the local region. We define *local dimensionality* of the entire data as the average dimensionality of the data over all local regions. The locality was controlled, in different experiments, using nearest neighbors (data density independent) as well as Euclidean distance (data density dependent) parameters and both gave similar results for our purpose.

In a synthetic example used to test the effectiveness of the tool, we used a quadratic curved surface embedded in three dimensions as shown in Fig.7.1(a). As can be seen, although the data is globally three dimensional, locally, it is a 2 dimensional surface. Using the dimensionality analysis tool, we computed the dimensionality of the data with varying locality. As shown in Fig.7.1(b), we can clearly observe a plateau at dimensionality 2 which shows that with the correct choice of locality, we can approximate the data distribution as patches of 2 dimensional manifolds.

©Sethu Vijayakumar 1998

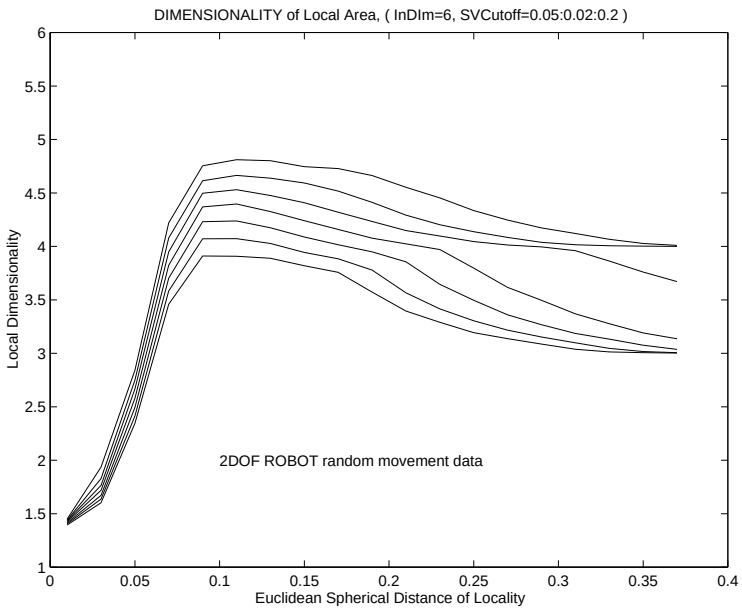


Figure 7.2: Dimensionality of 2 DOF robot movement data performing random motion in 6 dimensional input space.

7.2.2 The dimensionality analysis results

Using the tool described in the previous section, we looked at the dimensionality of input data from two different sources.

First, we used movement data from a 2DOF robot arm, which essentially means we have an original input space corresponding to 6 dimensions; the angular position, velocity and acceleration corresponding to each of the shoulder and elbow joints. The result of the analysis on the first set of data, 45,000 data points collected from random movements in the workspace is shown in Fig.7.2. The multiple lines show analysis done over a range of SVD cutoff thresholds. The result proves that on the average, the dimensionality of the data is around 4 dimensions, where the dimensionality of the data is defined as the average dimensionality over all the local regions, as explained in the previous subsection.

In the second set of experiments, we used data from the same robot arm but this time, the robot did a specific task of drawing an 'eight' in specified region of the workspace. It is interesting to note (see Fig.7.3) that the dimensionality has further dropped to around 3 dimensions on the average when executing specific tasks.

Next, we used 6000 data points from a 7DOF anthropomorphic robot arm, which is essentially a 21 dimensional input corresponding to the angular position, velocity and acceleration of the seven joints. The results of the analysis are shown in Fig.7.4.

©Sethu Vijayakumar 1998

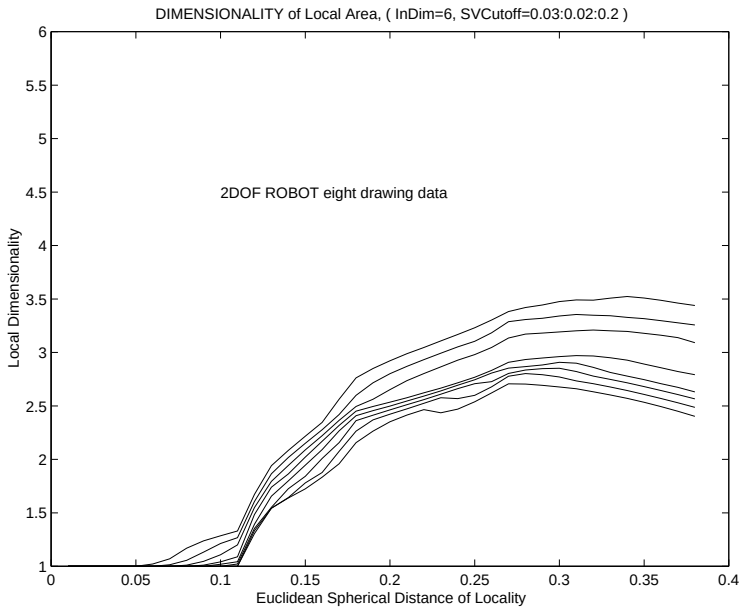


Figure 7.3: Dimensionality of 2 DOF robot movement data performing doing a specific task (here, drawing a figure "eight") in 6 dimensional input space. Note the further reduction in local dimensionality compared to random movement tasks.

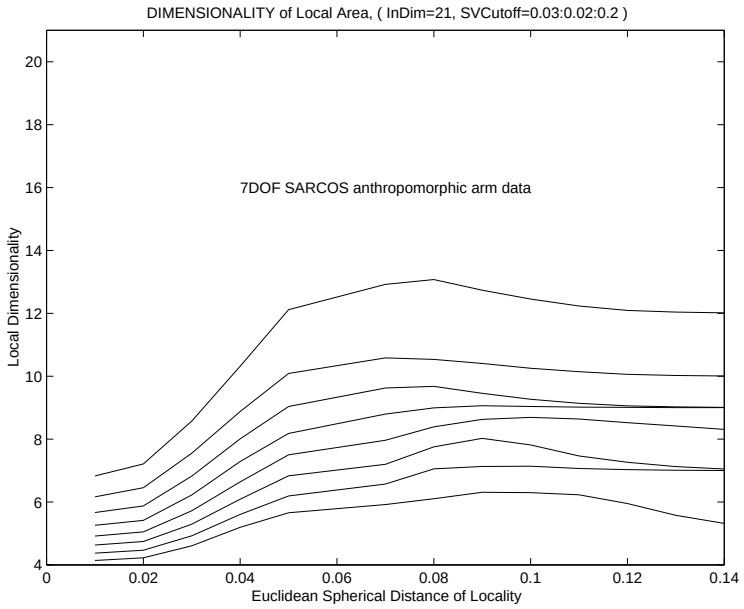


Figure 7.4: Dimensionality of 7 DOF SARCOS anthropomorphic robot arm data. A drastic reduction in dimensionality from the 21 dimensional input space to about 10-12 dimensions locally is observed.

©Sethu Vijayakumar 1998

It clearly shows that locally the input dimensions is much lower than 21 dimensions, in fact, on the average the local dimension works out to be no more than between 10-12 dimensions.

The results from these analysis show that these high dimensional problems can be handled in much lower dimensions locally if we can combine an effective method of finding the right projections (or important variance directions) locally and compute regression on top of this, preferably, in an incremental manner. They also provide a strong evidence towards the fact that internal models may actually be learned by biological systems without having to deal with the level of complexity considered before. This is not to say that physics does not make use of the entire dimensionality of the input data; rather it shows that locally the control or learning occurs employing much less number of dimensions.

In the next section, we look briefly at various candidates using which we can perform local dimensionality reduction (especially for regression) and do a Monte-Carlo study about their performance under different conditions.

7.3 Comparison of local dimensionality reduction techniques for regression

As mentioned in the previous section, learning in high dimensional spaces can be carried out using local learning techniques which exploit the local low dimensional distribution of input data. In pursuit of the question of what is the "right" method to perform local dimensionality reduction for regression [SVA97], this section will compare several candidate techniques under

1. perfectly fulfilled statistical prerequisites (e.g., Gaussian noise, Gaussian input distributions, perfectly linear data), and

2. less perfect conditions (e.g., non-Gaussian distributions, slightly quadratic data, incorrect guess of the dimensionality of the true data distribution).

We will focus on locally weighted regression (LWR) for nonlinear function approximation, details of which will be provided in Chapter 8, as it allows us to adapt a variety of global linear dimensionality reduction techniques and also because LWR has also found widespread application in several local learning systems [AMS97, JJ94, XJH95].

In particular, we will investigate locally weighted principal component regression (LWPCR), locally weighted joint data principal component analysis (LWPCA), lo-

©Sethu Vijayakumar 1998

cally weighted factor analysis (LWFA), and locally weighted partial least squares (LWPLS). Section 7.3.1 explains the data generative model that we will use for this comparison, while Sections 7.3.2 $\sim$ 7.3.5 will briefly outline these methods and their theoretical foundations. Section 7.3.6 will empirically evaluate the robustness of these methods using synthetic data sets that increasingly violate some of the statistical assumptions of the techniques.

7.3.1 Model used for comparison

The regression problems considered here assume that data originate from a process generating two sets of observable variables, the "inputs" $\check{\mathbf{x}}$ and the "outputs" $\check{y}$. The characteristics of the process ensures a functional relation

$$\check{y} = f(\check{\mathbf{x}}). \tag{7.3}$$

Both $\check{\mathbf{x}}$ and $\check{y}$ are observed through some measurement device that adds independent mean zero noise of different magnitude in each observable, such that

$$\mathbf{x} = \check{\mathbf{x}} + \epsilon_{\mathbf{x}} \tag{7.4}$$

$$y = \check{y} + \epsilon_y. \tag{7.5}$$

For the sake of simplicity, we will only focus on one-dimensional output data and functions f that are either linear or slightly quadratic, as these cases are the most common in nonlinear function approximation with locally linear models. Locality of the regression is ensured by weighting the error of each data point with a weight from a Gaussian kernel as

$$w_i = exp(-0.5(\mathbf{x}_i - \mathbf{x}_c)^T \mathbf{D}(\mathbf{x}_i - \mathbf{x}_c)), \tag{7.6}$$

where $\mathbf{x}_c$ denotes the center of the weighting kernel and $\mathbf{D}$ is a non-negative definite distance metric which determines the size and shape of the neighborhood contributing to the regression [AMS97]. Without loss of generality, all our data sets will set $\mathbf{x}_c$ to the zero vector, compute the weights, and then translate the input data such that the locally weighted mean,

$$\overline{\mathbf{x}} = \sum_{i=1}^{M} w_i \mathbf{x}_i / \sum_{i=1}^{M} w_i, \tag{7.7}$$

is zero, where M is the number of data points falling within the local region. The output data is similarly translated to be mean zero. Mean zero data is necessary for most of techniques considered below. The (translated) input data is summarized in the rows of the matrix $\mathbf{X}=[\mathbf{x}_1\ \mathbf{x}_2\ ...\mathbf{x}_M]^T$, the corresponding (translated) outputs are the elements of the vector $\mathbf{y}$, and the corresponding weights, determined from eq.(7.6) are in the diagonal matrix $\mathbf{W}$. In some cases, we need the joint input and output data, denoted as $\mathbf{Z} = [\mathbf{z}_1\ \mathbf{z}_2 ... \mathbf{z}_M]^T = [\mathbf{X}\ \mathbf{y}]$.

©Sethu Vijayakumar 1998

7.3.2 Locally weighted factor analysis(LWFA)

Factor analysis [Eve84] is a technique of dimensionality reduction which is the most appropriate since it tries to base the estimation on the basis of how we generate our regression data, i.e., in other words through *latent* or hidden variables. It assumes the observed data $\mathbf{z}$ was produced by a mean zero independently distributed k-dimensional vector of factors $\mathbf{v}$, transformed by the matrix $\mathbf{U}$, and contaminated by mean zero independent noise ϵ with diagonal covariance matrix $\mathbf{\Omega}$:

$$\mathbf{z} = \mathbf{U}\mathbf{v} + \epsilon, \tag{7.8}$$

where

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ y \end{bmatrix}, \tag{7.9}$$

$$\epsilon = \begin{bmatrix} \epsilon_x \\ \epsilon_y \end{bmatrix}, \tag{7.10}$$

$$E\{\epsilon\epsilon^T\} = \mathbf{\Omega}, \tag{7.11}$$

and $E\{.\}$ denotes the expectation operator. If both $\mathbf{v}$ and ϵ are normally distributed, the parameters $\mathbf{\Omega}$ and $\mathbf{U}$ can be obtained iteratively by the Expectation-Maximization algorithm (EM) [RT82]. For a linear regression problem, one assumes that $\mathbf{z}$ was generated with

$$\mathbf{U} = [\mathbf{I}\ \beta]^T \tag{7.12}$$

$$\mathbf{v} = \check{\mathbf{x}} \tag{7.13}$$

where β denotes the vector of regression coefficients of the linear model

$$y = \beta^T \mathbf{x}$$

and $\mathbf{I}$ the identity matrix. After calculating $\mathbf{\Omega}$ and $\mathbf{U}$ using EM algorithm in joint data space as formulated in eq.(7.8), an estimate of β can be derived from the conditional probability $P(y|\mathbf{x})$. As all distributions are assumed to be normal, the expected value of y is the mean of this conditional distribution. Let $\mathbf{Z} = [\mathbf{z}_1\ \mathbf{z}_2 ... \mathbf{z}_M]^T$ and $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2 ... \mathbf{v}_M]^T$. The locally weighted version (LWFA) of β can be obtained together with an estimate of the factors $\mathbf{v}$ from the joint weighted covariance matrix Ψ of $\mathbf{z}$ and $\mathbf{v}$ as

$$E\{\begin{bmatrix} y \\ \mathbf{v} \end{bmatrix} |\mathbf{x}\} = \begin{bmatrix} \beta^T \\ \mathbf{B} \end{bmatrix} \mathbf{x} = \Psi_{21}\Psi_{11}^{-1}\mathbf{x}, \tag{7.14}$$

where

$$\Psi = [\mathbf{Z}^T\ \mathbf{V}^T]\mathbf{W}\begin{bmatrix} \mathbf{Z} \\ \mathbf{V} \end{bmatrix} / \sum_{i=1}^{M} w_i = \begin{bmatrix} \Omega + \mathbf{U}\mathbf{U}^T & \mathbf{U} \\ \mathbf{U}^T & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \Psi_{11} & \Psi_{12} \\ \Psi_{21} & \Psi_{22} \end{bmatrix}, \tag{7.15}$$

©Sethu Vijayakumar 1998

and $\mathbf{B}$ is a matrix of coefficients involved in estimating the factors $\mathbf{v}$. Note that unless the noise ϵ is zero, the estimated β is different from the true β as it tries to average out the noise in the data.

7.3.3 Locally weighted principal component regression(LWPCR)

Although not optimal, a computationally efficient techniques of dimensionality reduction for linear regression is the principal component regression(LWPCR)[Mas65]. Let $\mathbf{C}^{lwpcr}$ represent the covariance matrix of the weighted input data written as

$$\mathbf{C}^{lwpcr} = \sum_{i=1}^{M} w_i(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T / \sum_{i=1}^{M} w_i \tag{7.16}$$

and the eigen value decomposition of $\mathbf{C}^{lwpcr}$ be given as

$$\mathbf{C}^{lwpcr}\mathbf{c}_i = \lambda_i \mathbf{c}_i \tag{7.17}$$

where, $\lambda_1 > \lambda_2 > ....$ The dimensionality reduction step in LWPCR projects the inputs onto the largest k principal components of the weighted covariance matrix, $\mathbf{C}_{lwpcr}$ by the transformation matrix $\mathbf{U}$ whose columns are the unit eigen vectors corresponding to the k largest eigen values of $\mathbf{C}^{lwpcr}$ given by

$$\mathbf{U} \equiv [\mathbf{c}_1 \ \mathbf{c}_2 \ ... \ \mathbf{c}_k]. \tag{7.18}$$

If β denotes the vector of regression coefficients of the linear model $\mathbf{y} = \beta^T\mathbf{x}$, then, β can be calculated as

$$\beta = (\mathbf{U}^T\mathbf{X}^T\mathbf{W}\mathbf{X}\mathbf{U})^{-1}\mathbf{U}^T\mathbf{X}^T\mathbf{W}y. \tag{7.19}$$

Eq.(7.19) is inexpensive to evaluate since after projecting $\mathbf{X}$ with $\mathbf{U}$, $\mathbf{U}^T\mathbf{X}^T\mathbf{W}\mathbf{X}\mathbf{U}$ becomes a diagonal matrix that is easy to invert. LWPCR assumes that the inputs have additive spherical noise, which includes the zero noise case. Since, during dimensionality reduction, LWPCR does not take into account the output data, it is endangered by clipping input dimensions with low variance which nevertheless have important contribution to the regression output. However, from a statistical point of view, it is less likely that low variance inputs have significant contribution in a linear regression, as the confidence bands of the regression coefficients increase inversely proportional with the variance of the associated input. If the input data has non-spherical noise, LWPCR is prone to focus the regression on irrelevant projections.

©Sethu Vijayakumar 1998

7.3.4 Locally weighted principal component analysis in joint data space (LWPCA)

An alternative way of determining the parameters in a reduced space employs locally weighted principal component analysis (LWPCA) in the joint data space. Let us denote the weighted covariance matrix of the joint data $\mathbf{Z} = [\mathbf{X}\ \mathbf{y}]$ as $\mathbf{C}^{lwpca}$, which is given by

$$\mathbf{C}^{lwpca} = \sum_{i=1}^{M} w_i(\mathbf{z}_i - \overline{\mathbf{z}})(\mathbf{z}_i - \overline{\mathbf{z}})^T / \sum_{i=1}^{M} w_i \tag{7.20}$$

and let the eigen value decomposition of $\mathbf{C}^{lwpca}$ be given as

$$\mathbf{C}^{lwpca}\mathbf{c}_i = \lambda_i \mathbf{c}_i \tag{7.21}$$

where, $\lambda_1 > \lambda_2 > ....$ Let us define a dimensionality reduction transformation matrix $\mathbf{U}$ as

$$\mathbf{U} \equiv [\mathbf{c}_1\ \mathbf{c}_2\ ...\ \mathbf{c}_{k+1}], \tag{7.22}$$

which is the projection of the joint data onto space spanned by the $k+1$ largest principal components of the joint data. Noting that the eigenvectors in $\mathbf{U}$ are unit length, the matrix inversion theorem of Horn & Johnson [HJ94] provides a means to derive an efficient estimate of β as

$$\beta = \mathbf{U}_x(\mathbf{U}_y^T - \mathbf{U}_y^T(\mathbf{U}_y\mathbf{U}_y^T - \mathbf{I})^{-1}\mathbf{U}_y\mathbf{U}_y^T), \tag{7.23}$$

where $\mathbf{U}$ is decomposed into $\mathbf{U}_x$ and $\mathbf{U}_y$ as

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_x \\ \mathbf{U}_y \end{bmatrix}, \tag{7.24}$$

and in our one dimensional output case, $\mathbf{U}_y$ is just a $(1 \times k)$-dimensional row vector. As a result, the evaluation of eq.(7.23) does not require a matrix inversion anymore but rather a division. If we assume normal distributions in all variables as in LWFA, LWPCA is the special case of LWFA where the noise covariance is spherical, i.e., the same magnitude of noise in all observables. Under these circumstances, the subspaces spanned by $\mathbf{U}$ in both methods will be the same. However, the regression coefficients of LWPCA will be different from those of LWFA unless the noise level is zero, as LWFA optimizes the coefficients according to the noise in the data (refer eq.(7.14)). Thus, for normal distributions and a correct guess of k, LWPCA is always expected to perform worse than LWFA.

7.3.5 Locally weighted partial least squares(LWPLS, LWPLS1)

Partial least squares [FF93] recursively computes orthogonal projections of the input data and performs single variable regressions along these projections on the residuals

©Sethu Vijayakumar 1998

of the previous iteration step. A locally weighted version of partial least squares (LWPLS) proceeds using the following pseudocode, where k is the dimension of the reduced space.

Initialize:

$$\begin{aligned} \mathbf{D}_0 &= \mathbf{X} \\ e_0 &= y \end{aligned}$$

For i=1 to k:

$$\mathbf{u}_i = \mathbf{D}_{i-1}^T \mathbf{W} e_{i-1} \tag{7.25}$$

$$\mathbf{s}_i = \mathbf{D}_{i-1} \mathbf{u}_i \tag{7.26}$$

$$\mathbf{p}_i = \mathbf{D}_{i-1}^T \mathbf{W} e_{i-1} / (\mathbf{s}_i^T \mathbf{W} \mathbf{s}_i) \tag{7.27}$$

$$\beta_i = \mathbf{s}_i^T \mathbf{W} e_{i-1} / (\mathbf{s}_i^T \mathbf{W} \mathbf{s}_i) \tag{7.28}$$

$$\mathbf{D}_i = \mathbf{D}_{i-1} - \mathbf{s}_i \mathbf{p}_i^T \tag{7.29}$$

A prediction with LWPLS cannot be summarized by a single vector multiplication as in LWFA and LWPCA, but rather needs another recursive algorithm, described by the following pseudocode. Let us assume that $\mathbf{x}_q$ is the query vector.

Initialize:

$$\mathbf{d}_0 = \mathbf{x}_q \tag{7.30}$$

$$y = 0 \tag{7.31}$$

For i=1 to k:

$$\mathbf{s}_i = \mathbf{d}_{i-1}^T \mathbf{u}_i \tag{7.32}$$

$$y = y + \Delta y \tag{7.33}$$

$$\Delta y = \beta_i \mathbf{s}_i \tag{7.34}$$

$$\mathbf{d}_i = \mathbf{d}_{i-1} - \mathbf{s}_i \mathbf{p}_i^T \tag{7.35}$$

As all single variable regressions are ordinary univariate least-squares minimizations, LWPLS makes the same statistical assumption as ordinary linear regressions that only output variables have additive noise, but input variables are noiseless. The choice of the projections $\mathbf{u}$, however, introduces an element in LWPLS that remains statistically largely unclear [FF93]. A peculiarity of LWPLS is that it also regresses the inputs of the previous step against the projected inputs $\mathbf{s}$ in order to ensure the orthogonality of all the projections $\mathbf{u}$. Moreover, since the projections $\mathbf{u}$ are computed from the covariance of input *and* output data, the algorithm fits essentially more parameters in k iterations than LWFA and LWPCA. We will address this issue in our empirical evaluations by comparing k-step LWPLS with 1-step LWPLS, abbreviated LWPLS1.

©Sethu Vijayakumar 1998

7.3.6 Monte Carlo evaluations for comparison

In order to evaluate the candidate methods, data sets with 5 inputs and 1 output were randomly generated. Each data set consisted of 2000 training points and 10000 test points, distributed either uniformly or nonuniformly in the unit hypercube. The outputs were generated by either a linear or quadratic function. Afterwards, the 5-dimensional input space was projected into a 10-dimensional space by a randomly chosen distance preserving linear transformation. Finally, Gaussian noise of various magnitudes was added to both the 10-dimensional inputs and one dimensional output. For the test sets, the additive noise in the outputs was omitted. Each regression technique was localized by a Gaussian kernel (refer eq.(7.6)) with a 10-dimensional distance metric $\mathbf{D}$ =10$\mathbf{I}$. The precise experimental conditions followed closely those suggested by Frank and Friedman[FF93]:

- Two(2) kinds of linear functions $y = \beta_{lin}^T \mathbf{x}$ for:
  - **(i)** β_{lin} =[1 1 1 1 1]^T.
  - **(ii)** β_{lin} =[1 2 3 4 5]^T.
- Two(2) kinds of quadratic functions $y = \beta_{lin}^T \mathbf{x} + \beta_{quad}^T [x_1^2\ x_2^2\ x_3^2\ x_4^2\ x_5^2]^T$ for:
  - **(i)** β_{lin} =[1 1 1 1 1]^T and β_{quad} =0.1[1 1 1 1 1]^T.
  - **(ii)** β_{lin} =[1 2 3 4 5]^T and β_{quad} =0.1[1 4 9 16 25]^T.
- Three(3) kinds of noise conditions, each with two(2) sub-conditions:
  - **(i)** only output noise:
    - **(a)** low noise: local signal/noise ratio (lsnr) = 20 and
    - **(b)** high noise: lsnr =2
  - **(ii)** equal noise in inputs and outputs:
    - **(a)** low noise : $\epsilon_{x,n} = \epsilon_y = N(0, 0.01^2),\ n = 1, 2, ..., 10$ and
    - **(b)** high noise : $\epsilon_{x,n} = \epsilon_y = N(0, 0.1^2),\ n = 1, 2, ..., 10$.
  - **(ii)** unequal noise in inputs and outputs:
    - **(a)** low noise : $\epsilon_{x,n} = \epsilon_y = N(0, 0.01^2),\ n = 1, 2, ..., 10$, lsnr=20 and
    - **(b)** high noise : $\epsilon_{x,n} = \epsilon_y = N(0, 0.1^2),\ n = 1, 2, ..., 10$, lsnr=2.
- Two(2) kinds of input distribution:
  - **(i)** uniform in unit hyper cube,
  - **(ii)** uniform in unit hyper cube excluding data points which activate a Gaussian weighting function eq.(7.6) at c = [0.5 0 0 0 0]^T with $\mathbf{D}$ =10*$\mathbf{I}$ more than $\mathbf{w}$ =0.2 (this forms a "hyper kidney" shaped distribution).

©Sethu Vijayakumar 1998

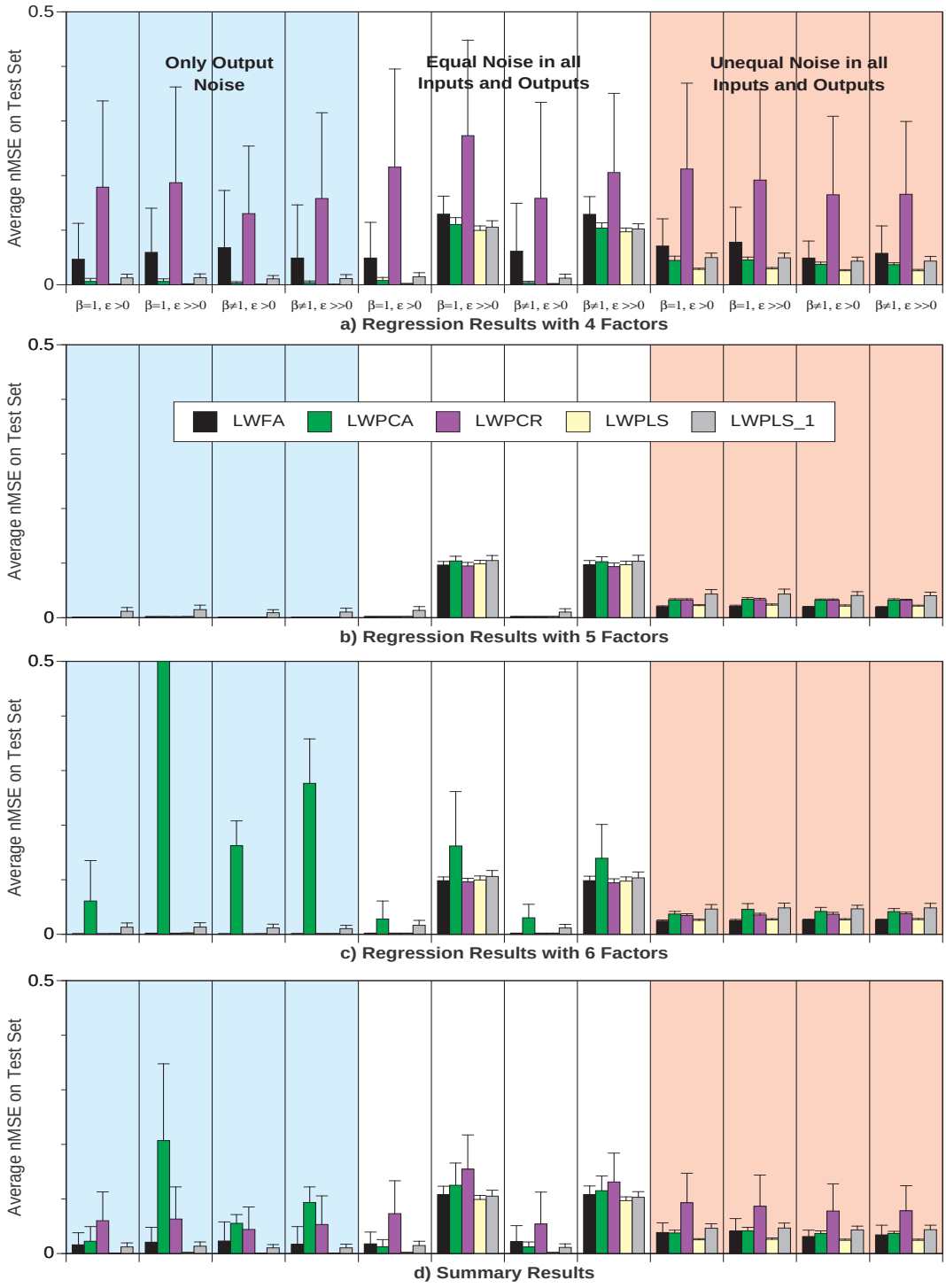


Figure 7.5: Average summary results of Monte Carlo experiments. Each chart is primarily divided into the three major noise conditions, cf. headers in chart (a). In each noise condition, there are four further subdivision: coefficients of linear or quadratic model are equal with (i) low added noise; (ii) with high added noise; coefficients of linear or quadratic model are different with (iii) low noise added; (iv) with high added noise. Refer to text and descriptions of Monte Carlo studies for further explanations.

©Sethu Vijayakumar 1998

Every algorithm was run[1] 30 times on each of the 48 combinations of the conditions. Additionally, the complete test was repeated for three further conditions varying the dimensionality, called factors in accordance with LWFA, that the algorithms assumed to be the true dimensionality of the 10-dimensional data from k=4 to 6, i.e., too few, correct, and too many factors. The average results are summarized in Fig.7.5. Fig.7.5(a), (b), (c) show the summary results of the three factor conditions. Besides averaging over the 30 trials per condition, each mean of these charts also averages over the two input distribution conditions and the linear and quadratic function condition, as these four cases are assumed to be normal violations of the statistical assumptions in nonlinear function approximation with locally linear models. In Fig.7.5(b), the number of factors equals the underlying dimensionality of the problem, and all algorithms are essentially performing equally well. For perfectly Gaussian distributions in all random variables (not shown separately), LWFA's assumptions are perfectly fulfilled and it achieves the best results, however, almost indistinguishable closely followed by LWPLS. For the *unequal noise condition*, the two PCA based techniques, LWPCA and LWPCR, perform the worst since, as expected, they choose noise contaminated projections. However, when violating the statistical assumptions, LWFA loses parts of its advantages, such that the summary results become fairly balanced in Fig.7.5(b).

The quality of function fitting changes significantly when violating the correct number of factors, as illustrated in Fig.7.5(a) and (c). For too few factors (Fig.7.5(a)), LWPCR performs worst because it randomly omits one of the principal components in the input data, without respect to how important it is for the regression. The second worse is LWFA: according to its assumptions it believes that the signal it cannot model must be noise, leading to degraded regression results. LWPLS has a clear lead in this test, closely followed by LWPCA and LWPLS1. For too many factors than necessary (Fig.7.5(c)), it is now LWPCA which degrades. This effect is due to its extracting one very noise contaminated projection which strongly influences the recovery of the regression parameters in eq.(7.23). All other algorithms perform almost equally well, with LWFA and LWPLS taking a small lead.

Fig.7.5(d) summarizes all the Monte Carlo experiments in a final average plot. Except for LWPLS, every other technique showed at least one clear weakness in one of our *robustness* tests. It was particularly an incorrect number of factors which made these weaknesses apparent. For high-dimensional regression problems, the local dimensionality, i.e., the number of factors, is not a clearly defined number but rather a varying quantity, depending on the way the generating process operates. Usually, this process does not need to generate locally low dimensional distributions,

[1] Except for LWFA, all methods can evaluate a data set in non-iterative calculations. LWFA was trained with EM for maximally 1000 iterations or until the log-likelihood increased less than $1.e^{-10}$ in one iteration.

©Sethu Vijayakumar 1998

however, it often "chooses" to do so, for instance, as human arm movements follow stereotypic patterns despite the fact that they could generate arbitrary ones. Thus, local dimensionality reduction needs to find autonomously the appropriate number of local factor. Locally weighted partial least squares turned out to be a surprisingly robust technique for this purpose, even outperforming the statistically appealing probabilistic factor analysis. As in principal component analyses, LWPLS's number of factors can easily be controlled just based on a variance-cutoff threshold in input space [FF93], while factor analysis usually requires expensive cross-validation techniques. Simple, variance-based control over the number of factors can actually improve the results of LWPCA and LWPCR in practice, since, as shown in Fig.7.5(a), LWPCR is more robust towards overestimating the number of factors, while LWPCA is more robust towards an underestimation.

Hence, if one is interested in dynamically growing the number of factors while obtaining already good regression results with too few factors, PCA based approaches seems to be a good solution. Especially, it will be shown later, that LWPCR based techniques are well suited for adapting to *incremental* approximations with *changing input distributions*, a scenario that we will encounter in the problems of robot sensorimotor map estimation. Therefore, we will concentrate on PCA based local dimensionality techniques in this work and in the next chapter, derive local, weighted and incremental versions for use on top of our regression framework.

7.4 Comments

In this chapter, we have looked at various dimensionality reduction techniques and looked at the strengths and weaknesses of a few of them. Most of the techniques discussed here are *second-order* methods, i.e., they only use information contained in the covariance matrix of the data vector $\mathbf{x}$. If the variable has a Gaussian or normal distribution, then, it's distribution is completely determined by this second-order information. Thus, it is useless to include any other information. A major reason for the popularity of second order methods is that they are computationally simple, requiring only classical matrix manipulations. Future research along the lines of using the output data information also (LWPCR, at present, uses only input data information for dimensionality reduction) is underway.

Higher order techniques are useful when one wants to look at properties of data that are not described using Gaussian or normal distributions, for example, clustering and independence of components. Some conventional methods based on higher-order statistics are projection pursuit, redundancy reduction and blind deconvolution. We have done some research into applying Independent Component Analysis(ICA) based techniques for dimensionality reduction but have not been able to get satisfactory results for the same.

©Sethu Vijayakumar 1998

©Sethu Vijayakumar 1998

8

Locally Weighted Learning with LASS

8.1 Introduction

In Part II of this dissertation, we had concentrated on exact methods of learning with global basis functions. However, as the complexity of the problem grew, we saw that it became increasing difficult to deal with certain essentials for the global learning methods. First, there is the problem of *model selection* which brings the bias-variance tradeoff [GBD92] into the forefront again. As the learning task gets more complex, it becomes unclear how to directly deal with the problem of selecting the right search space or providing the right bias; this, in our global approximation scheme, corresponds to deciding on the correct Hilbert space to search in. Second, we would like to address an incremental learning scenario in which not only the parameters of the learning system are updated incrementally but also in which a new data point is *discarded* after being incorporated into the learning scheme and hence, cannot be re-used. Also, we consider cases in which the input-output distributions are unknown and in which these distributions may change over time. Third, there is the problem of *negative interference*, the forgetting of useful knowledge while focusing on learning from new data. Methods to prevent these undesirable effects usually require either a validation set, memorizing all the training data or strong prior knowledge of the learning problem. However, none of the alternatives are available in the setting we have described as we want to avoid storing data and do not have knowledge about the structure of the learning task. Finally, we have to handle cases in which there may be many redundant, noisy or unrelated inputs to the learning system. This situation

resembles the learning of sensory and sensorimotor transformations in biology, and it also applies to a variety of artificial domains ranging from autonomous robotic systems to process control.

In order to address and overcome these problems, we will make use of non-parametric regression statistics to perform local incremental learning using receptive fields. Learning with spatially localized basis functions has become a popular paradigm in machine learning and neurobiological modeling. In the context of radial basis function networks [MD88, PG90], it was demonstrated that these learning methods offer an alternative to learning with global basis functions and that their theoretical foundations can be solidly grounded in approximation theory. In neurophysiological studies, the concept of localized information processing in form of receptive fields have been known and a wealth of experimental evidence have been collected which suggests that information processing using local receptive fields is a ubiquitous organizational principle in neurobiology [OF96, DD95].

We will develop, in stages, an algorithm for handling these requirements. Summarizing how we are going to specifically address each of the above mentioned problems:

1. The *model selection* problem, in other words, allocation of the appropriate number of resources (e.g. receptive fields) for a particular problem is circumvented by using a **non-parametric regression** scheme which **incrementally allocates resources** based on the complexity encountered (data driven).

2. The ability to *discard* data after incorporation is achieved by a unique incremental learning scheme which provides a **consistent incremental update** which approximates the batch learning results using only the **current data** and the learned function. The changing input and output distributions are handled using a **forgetting** factor.

3. The problem of *negative interference* is circumvented because each of the local receptive field is trained entirely **independently** of all other receptive fields and all the local parameters are learned **without competition** and without the need for any kind of communication.

4. The problem of handling *redundant* or *irrelevant inputs* is achieved by using an **unsupervised** preprocessing stage for **dimensionality reduction** which projects input data onto a low dimensional subspace before performing the regression analysis. For this purpose, we will use a locally weighted and incremental PCA based dimensionality reduction technique since the analysis of Chapter 7 showed that they are simple and competent tools for the task.

The detailed description of the implementation of each of these solutions will be carried out in the course of introducing the **L**ocal **A**daptive **S**ub**S**pace Regres-

©Sethu Vijayakumar 1998

sion(LASS) algorithm [VS97] in the later half of this chapter. LASS is a learning algorithm that exploits low dimensional dense distributions in high dimensional data by combining a dynamically growing local dimensionality reduction technique as a preprocessing step with a nonparametric learning scheme, local weighted regression[AMS97], that learns the region of validity of regression. Section 8.2 describes the underlying statistical model along with the parameters that has to be learned in LASS. Sections 8.3 $\sim$ 8.5 specifically outline how these parameters can be learned incrementally, locally and without competition. Sections 8.6 and 8.7 outlines the working of the LASS algorithm and provides the implementation pseudocode, respectively.

8.2 The statistical model of LASS

The assumed underlying statistical model of our problem is the standard regression model

$$y = f(\mathbf{x}) + \epsilon, \tag{8.1}$$

where $\mathbf{x} \in R^N$ denotes the N dimensional input vector, y, without loss of generality, a scalar output and $f(.)$ a deterministic function mapping the input $\mathbf{x}$ to the noise free output $y-\epsilon$. The additive random noise ϵ is assumed to be mean zero, $E\{\epsilon\} = 0$ but otherwise of unknown distribution. Here $E(.)$ denotes the expectation operator. The input data is distributed according to the density $p(\mathbf{x})$.

LASS consists of an automatically adjusting number of elements, each of which is processing data in the same way. The overall structure of LASS is shown in Fig.8.1. We will associate a subscript k with each local element of LASS. All the local elements of LASS receive the same set of N dimensional input $\mathbf{x}$ and use it to make their local predictions $\hat{y_k}$. LASS combines these local predictions $\hat{y_k}$ to produce a weighted final prediction $\hat{y}$ as shown in Fig.8.1. Zooming in, the different stages and additional notation of the information flow in one element of the LASS system are shown in Fig.8.2. In essence, the input is transformed into a predicted output $\hat{y_k}$ by two linear transformations, $\mathbf{W}_k^{PCA}$ and $\mathbf{b}_k$. $\mathbf{W}_k^{PCA}$ performs the dimensionality reduction and $\mathbf{b}_k$ fits a (hyper)plane to the reduced data. Additionally, a weight w_k is calculated with the help of the connections $\mathbf{D}_k$. Based on a Gaussian activation function, the weight, w_k, is computed for every training data point $(\mathbf{x}, y)$ as

$$w_k = exp(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_k)^T\mathbf{D}_k(\mathbf{x}-\mathbf{c}_k)), \tag{8.2}$$
$$\text{where } \mathbf{D}_k = \mathbf{M}_k^T\mathbf{M}_k \text{ for positive definiteness.}$$

w_k indicates how much this LASS element should contribute to the total prediction of the entire system. The center of the Gaussian $\mathbf{c}_k$ is assigned at the time of creation

©Sethu Vijayakumar 1998

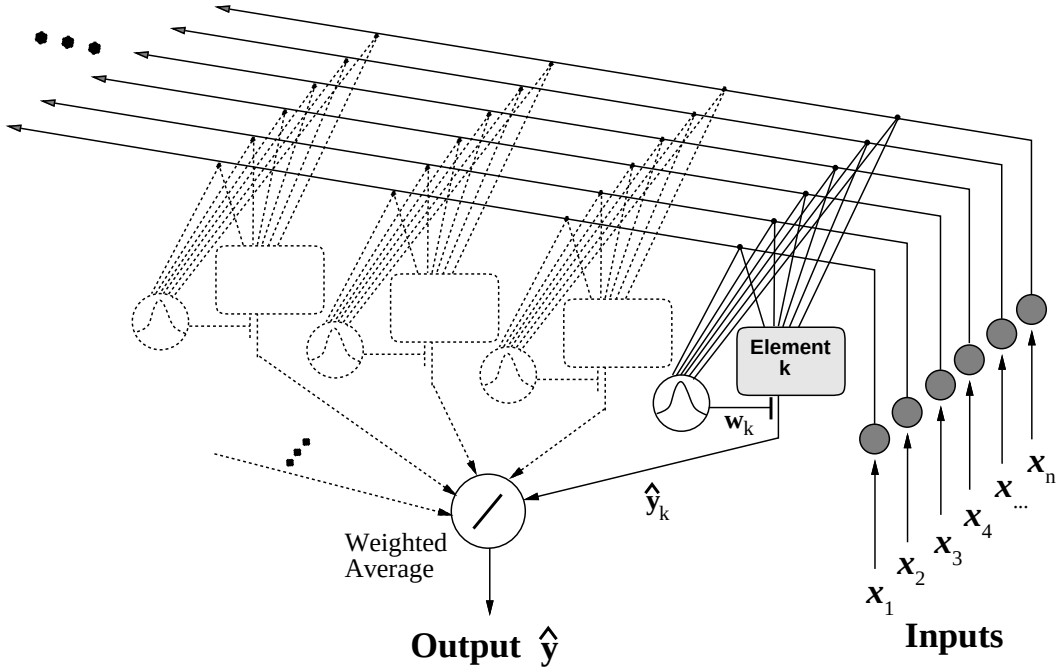


Figure 8.1: Organization of local elements in LASS

of the LASS unit and remains stationary. The matrix $\mathbf{D}_k$, referred to as the distance metric, determines the size and shape of the "receptive field" created by eq.(8.2). The total prediction $\hat{y}$ of the entire LASS system results from the weighted average of the individual predictions $\hat{y}_k$ of all the K elements:

$$\hat{y} = (\sum_{k=1}^{K} w_k \hat{y}_k)/(\sum_{k=1}^{K} w_k). \tag{8.3}$$

From now on, we will drop the subscript k since every LASS elements learns independently of every other one and is updated by the same formulae.

We will summarize and list out the notations that will be used with regard to the LASS learning algorithm. Each of the LASS processing elements maintains these set of running variables independent of each other.

c : The center of the receptive field.

x : The input vector of N dimensions.

$\mathbf{x}_{reg}$**:** The vector of reduced dimension L to be used for regression analysis.

$\overline{\mathbf{x}}$**:** The mean of the M weighted input data vectors.

$\mathbf{x}_{mz}$**:** The mean subtracted input data, $\mathbf{x} - \overline{\mathbf{x}}$.

©Sethu Vijayakumar 1998

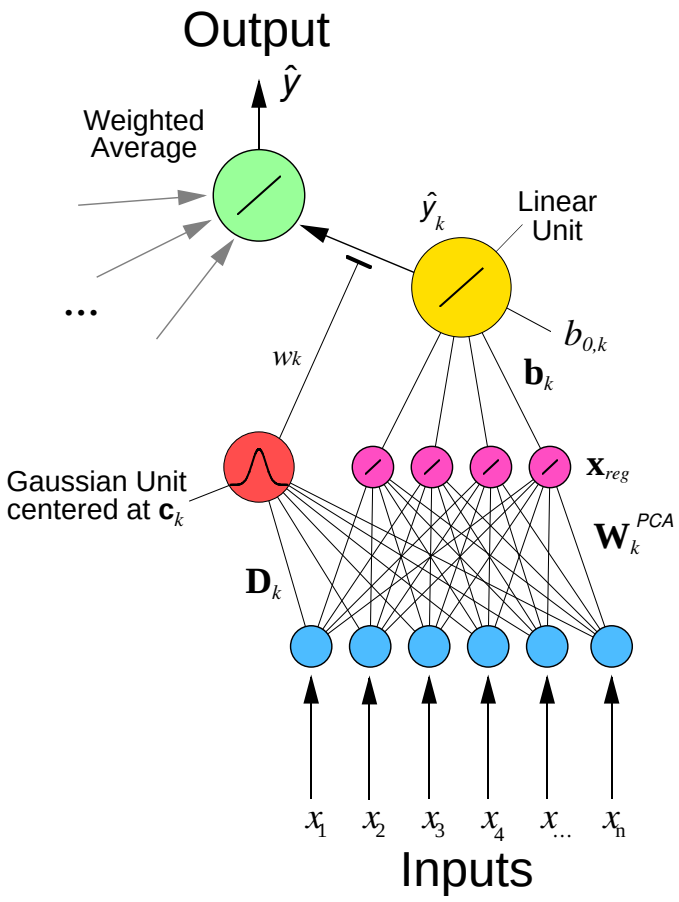


Figure 8.2: Illustration of the information processing stages in one element of LASS.

$\mathbf{W}^{PCA}$**:** The $L \times N$ matrix performing the PCA projection of the N dimensional vector $\mathbf{x}$ onto the L dimensional principal subspace (to be defined shortly).

b: Regression parameter which fits a (hyper)plane to the data.

D: The distance metric which determines the size and shape of the receptive field.

There are three variable parameters to be learned (adjusted) during the course of training. They are the transformation $\mathbf{W}^{PCA}$, which projects the local weighted input data onto a lower dimensional subspace, the local regression coefficient **b** which fits a (hyper)plane using the reduced dimensional data, and the distance metric parameter **D** which controls the region of validity of a local model. The next section will, in turn, detail out how each of these parameters are learned in an incremental manner in our learning model.

©Sethu Vijayakumar 1998

8.3 Locally weighted dimensionality reduction in LASS (LWPCA)

Various candidates can be considered for dimensionality reduction in order to exploit locally low dimensional data distributions as discussed in Chapter 7. From choices ranging from principal component analysis (PCA), independent component analysis (ICA), partial least squares (PLS) and factor analysis, we employ locally weighted PCA (LWPCA). Monte Carlo evaluations in Chapter 7 and work by Frank & Friedman [FF93] have shown that although Principal Component Analysis based Regression (PCR) performs dimensionality reduction only in the input space, it is quite competitive with more sophisticated statistical dimensionality reduction techniques. Moreover, LWPCA offers a good compromise in terms of computational feasibility and numerical robustness.

In this section, we explain the fundamentals of PCA in more detail and derive methods of computing a weighted incremental version using an optimization criterion based on *optimal reconstruction* of compressed signals. We will go on to derive an incremental update formula for incrementally learning the projection operator $\mathbf{W}^{PCA}$.

8.3.1 The basic theory

Principal Component Analysis or PCA is widely used in signal processing, statistics and neural networks. In some application areas, it is also called the (discreet) Karhunen-Loève transform, or the Hotelling transform.

The basic goal of PCA dimensionality reduction is to locally project the original N dimensional input vector $\mathbf{x}$ onto a L dimensional principal subspace, a subspace which accounts for the maximum variance of the data in L dimensions. In principle, the transformation from the higher to lower dimension, represented by the $L \times N$ matrix $\mathbf{W}^{PCA}$, can be analytically derived using the covariance matrix of the input data as shown below. Let $\mathbf{C}$ represent the covariance matrix of the weighted mean zero input data written as

$$\mathbf{C} = \sum_{i=1}^{M} w_i \mathbf{x}_{mz,i} \mathbf{x}_{mz,i}^T / \sum_{i=1}^{M} w_i. \tag{8.4}$$

where

$$\mathbf{x}_{mz,i} = \mathbf{x}_i - \overline{\mathbf{x}} \tag{8.5}$$

is the mean subtracted input. The eigen value decomposition of $\mathbf{C}$ can be written

©Sethu Vijayakumar 1998

as

$$\mathbf{C}\mathbf{c}_i = \lambda_i \mathbf{c}_i \tag{8.6}$$

where, $\lambda_1 > \lambda_2 > ....$ Then, $\mathbf{W}^{PCA}$ can be expressed using the L largest principal components of the weighted covariance matrix as

$$\mathbf{W}^{PCA} \equiv [\mathbf{c}_1 \ \mathbf{c}_2 \ ... \ \mathbf{c}_L]. \tag{8.7}$$

Hence, the projections of the data onto the lower dimensional principal subspace can be found using

$$\mathbf{x}_{reg} = \mathbf{W}^{PCA}\mathbf{x}_{mz}. \tag{8.8}$$

$\mathbf{x}_{reg}$ is the dimension reduced data on which we will perform the regression analysis. It is important to note that from the view of optimal reconstruction or data compression, it is not necessary to use the L principal components themselves, rather we can use any other orthonormal basis of the subspace spanned by the principal components (called the PCA subspace)[OOW92]. However, the approach suggested here can be used only for batch operations, i.e., we need to accumulate all data and compute it's covariance matrix and perform the projections in a single batch operation. In the next subsection, we will show how to obtain the PCA of a *locally weighted* data set *incrementally.*

8.3.2 Locally weighted and incremental implementation

Here we derive an update formula for incrementally computing the LWPCA projection matrix, $\mathbf{W}^{PCA}$, for a locally weighted data set. Let $\mathbf{x}_m$ represent the m-th input data. Correspondingly, let $\mathbf{x}_{mz,m}$ represent the mean subtracted m-th input data and $\mathbf{x}_{reg,m}$ represent the output of the dimensionality reduction stage for the m-th input data. Let us assume that the data is weighted using a Gaussian weighting kernel shown in eq.(8.2) to provide a weight w_m corresponding to each mean centered data point, $\mathbf{x}_{mz,m}$. Then, we can derive an incremental update rule for $\mathbf{W}^{PCA}$ by minimizing a weighted cost criterion in the spirit of Minimum Description Length (MDL) [Ris89]:

$$J_{PCA}[\mathbf{W}^{PCA}] = \frac{1}{2}\sum_{m=1}^{M} w_m \|\mathbf{x}_{reconst,m} - \mathbf{x}_{mz,m}\|^2, \tag{8.9}$$

where the variable $\mathbf{x}_{reconst,m}$ is calculated from the inverse transformation

$$\mathbf{x}_{reconst,m} = \mathbf{W}^{PCA^{-1}}\mathbf{x}_{reg,m} = \mathbf{W}^{PCA^T}\mathbf{x}_{reg,m}. \tag{8.10}$$

This holds since the update for $\mathbf{W}^{PCA}$ enforces that $\mathbf{W}^{PCA^T}$ is an orthogonal transformation and each of it's row has unit length. The minimization of eq.(8.9) is

©Sethu Vijayakumar 1998

achieved by gradient descent on the transformation matrix $\mathbf{W}^{PCA}$ with learning rate η as follows. Let $\mathbf{x}_{mz} \equiv (x_{mz,i})$ and $\mathbf{x}_{reg} \equiv (x_{reg,i})$ represent the mean subtracted vector and the output vector of the dimensionality reduction stage corresponding to the current input vector $\mathbf{x}$ (in the incremental update stage). Let w represent the weight computed for this input data $\mathbf{x}$. The update for $\mathbf{W}^{PCA}$ is given as

$$W_{ij}^{PCA^{(n+1)}} = W_{ij}^{PCA^{(n)}} + \eta \frac{\partial J_{PCA}}{\partial W_{ij}^{PCA}}. \tag{8.11}$$

where

$$\frac{\partial J_{PCA}}{\partial W_{ij}^{PCA}} = w x_{reg,i} (\sum_{r=1}^{i} x_{reg,r} W_{ij}^{PCA^n} - x_{mz,j}). \tag{8.12}$$

This corresponds to a weighted version of the incremental PCA algorithm of Oja [Oja82] and Sanger[San89]. In order to speed up learning, we can also derive a second order gradient descent minimization of eq.(8.9) based on [Sut92] by assigning an adjustable learning rate η_{ij} to each coefficient W_{ij}^{PCA}. Then, the second order update rule becomes

$$W_{ij}^{PCA^{(n+1)}} = W_{ij}^{PCA^{(n)}} + \eta_{ij} \frac{\partial J_{PCA}}{\partial W_{ij}^{PCA}}. \tag{8.13}$$

The individual learning rates η_{ij} is updated according to

$$\eta_{ij}^{(n+1)} = exp(\tau_{ij}^{(n+1)}), \tag{8.14}$$

where $\tau_{ij}^{(n+1)}$ is defined as

$$\tau_{ij}^{(n+1)} = \tau_{ij}^{(n)} - \theta \frac{\partial J_{PCA}}{\partial W_{ij}^{PCA}} h_{ij}^{(n)}. \tag{8.15}$$

Here, $h_{ij}^{(n+1)}$ is updated incrementally as

$$h_{ij}^{(n+1)} = h_{ij}^{(n)} [1 - \eta_{ij}^{(n+1)} \frac{\partial^2 J_{PCA}}{\partial W_{ij}^{PCA^2}}]^+ - \eta_{ij}^{(n+1)} \frac{\partial J_{PCA}}{\partial W_{ij}^{PCA}}, \tag{8.16}$$

where

$$[z]^+ = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}.$$

Computing this second order update adds only little computation. It adjusts the learning rates η_{ij} in geometric steps by gradient descent in the meta parameter τ_{ij} with meta learning rate θ. More details are given in [Sut92]. For the calculation of $\mathbf{x}_{mz}$, it is also necessary to incrementally obtain the weighted mean of $\mathbf{x}$ by

$$\bar{\mathbf{x}}^{(n+1)} = (W^{(n)} \bar{\mathbf{x}}^{(n)} + w\mathbf{x})/(W^{(n)} + w) \tag{8.17}$$

where $W^{(n)}$ is a cumulative sum of the receptive field weights updated as $W^{(n+1)} = W^{(n)} + w$. The final output from the dimensionality reduction stage of LASS is the L dimensional projected data $\mathbf{x}_{reg}$.

©Sethu Vijayakumar 1998

8.4 Locally weighted regression (LWR) in LASS

This section describes the processing of the input data which follows after the dimensionality reduction has been carried out in Section 8.3. Each of the LASS elements (refer Fig.8.2) performs a local regression on the projected reduced dimensional data $\mathbf{x}_{reg}$ to make a local prediction as

$$\hat{y} = \mathbf{x}_{reg}^T \mathbf{b} + b_0 \equiv \tilde{\mathbf{x}}^T \beta, \tag{8.18}$$

where

$$\tilde{\mathbf{x}} = (\mathbf{x}_{reg}^T, 1)^T. \tag{8.19}$$

This can be interpreted as fitting a local (hyper)plane through the data in the least squared error sense in the region of validity of the receptive field. A one-dimensional function fitting example using local linear models is illustrated in Fig.8.3. The Gaussian weighting kernels shown at the bottom of the figure determines the shape and size of the region of validity of each receptive field. Local linear models corresponding to each receptive field fits a line through the data to perform a piecewise linear approximation of the highly non-linear function of Fig.8.3.

A fast and efficient calculation of β of eq.(8.18) can be performed by locally weighted regression as shown in Atkeson, Moore & Schaal[AMS97]. The objective function of locally weighted regression is the minimization of the weighted mean squared error criterion

$$J_{REG}[\beta] = \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} w_i \, (y_i - \hat{y}_i)^2, \tag{8.20}$$

where y_i represents the true output of the i-th training data and $\hat{y}_i$ is it's approximation by LASS. It will be useful to leave the incremental learning framework for a moment and think in terms of a batch update of β. If we summarize the input part of all M training points $\tilde{\mathbf{x}}$ in the rows of the matrix $\mathbf{X} = (\tilde{\mathbf{x}}_1, ..., \tilde{\mathbf{x}}_M)^T$, the corresponding output part in the rows of the vector $\mathbf{Y} = (y_1, ..., y_M)^T$, and the corresponding weights in the diagonal matrix $\mathbf{W} = diag(w_1, ..., w_M)$, the regression parameters β can be calculated from a weighted regression as

$$\beta = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{Y} = \mathbf{P} \mathbf{X}^T \mathbf{W} \mathbf{Y}. \tag{8.21}$$

An incremental update for β is obtained by recursive least squares [LS86, SA96]. It results in *exactly* the same estimation for β as eq.(8.21) after one sequential sweep through the training data and constitutes a very fast Newton training method with guaranteed convergence to the global minimum of eq.(8.20).

However, it should be noted that the LWPCA pre-processing step does not only yield a computational advantage in terms of providing a significantly reduced input

©Sethu Vijayakumar 1998

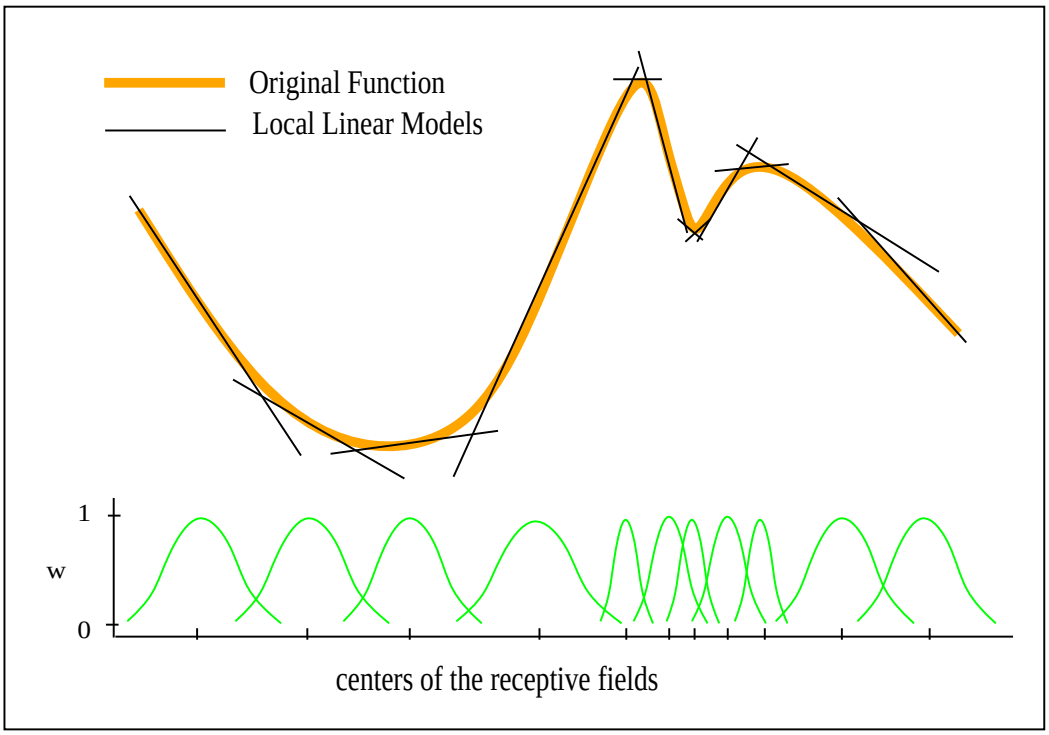


Figure 8.3: An example of approximation of a complex function using receptive field based locally linear models

to the regression; LWPCA also decorrelates the dimensions of $\mathbf{x}_{reg}$ Thus, the regression decomposes into $L+1$ univariate additive regressions [HT90], reducing the computational complexity of the regression from being quadratic in the number of regression inputs to being linear. As a result of this, the regression can be computed incrementally by merely maintaining a cumulative sum of the variance of $\tilde{\mathbf{x}} \equiv (\tilde{x}_i)$ represented by $\mathbf{s}_{xx} \equiv (s_{xx,i})$ and also the covariance of $\tilde{\mathbf{x}}$ with y, represented by $\mathbf{s}_{xy} \equiv (s_{xy,i})$ as

$$\beta_i^{(n+1)} = s_{xy,i}^{(n+1)} / s_{xx,i}^{(n+1)}, \tag{8.22}$$

where

$$s_{xy,i}^{(n+1)} = \lambda s_{xy,i}^{(n)} + w\tilde{x_i}y \tag{8.23}$$

and

$$s_{xx,i}^{(n+1)} = \lambda s_{xx,i}^{(n)} + w\tilde{x_i}^2. \tag{8.24}$$

The variable λ in eqs.(8.23) and (8.24) denotes a forgetting factor, a standard technique in recursive estimation [LS86], such that initial input to the regression, stemming from an LWPCA which has not properly converged yet, will not negatively influence the regression result in the long run. This is particularly useful in the case when the input distribution $p(\mathbf{x})$ is changing, which is very likely to happen

©Sethu Vijayakumar 1998

in incremental learning tasks. Imagine a robot learning the dynamics model of it's arm, a model which maps joint positions, joint velocities and joint accelerations to corresponding joint torques. Whenever the robot moves, it will receive valid data about this functional relationship. But since the robot is fulfilling different tasks at different times, the sampled data will come from quite different input distributions – for example, think of the difference between movements for cooking and movements for playing tennis. Under these circumstances, the projection direction and the projection subspace will have to be adapted with respect to the changing input distribution in order to reflect the true projection directions at the end, and subsequently, the regression parameter has to reflect these corrections, an effect which is implemented by using the forgetting factor along with the incremental update derived here.

8.5 Distance metric adaptation in LASS

The linear model represented by each LASS unit is valid only in the locality specified by the distance metric $\mathbf{D}$. In order to adapt to local differences in the spatial frequency of the outputs, we have to optimize the size and shape of the receptive fields in the local models by adapting the distance metric $\mathbf{D} = \mathbf{M}^T\mathbf{M}$.

At a first glance, one might hope that this can be done by gradient descent in $\mathbf{M}$ of the weighted mean squared error criterion

$$J'[\mathbf{M}] = \frac{1}{W}\sum_{i=1}^{M} w_i(y_i - \hat{y}_i)^2 \tag{8.25}$$

where

$$W = \sum_{i=1}^{M} w_i. \tag{8.26}$$

This is the basis of solution of locally weighted regression [AMS97] in eq.(8.21). However, minimizing eq.(8.25) may result in a quite inappropriate solution. If for each training point, one receptive field is centered right on this point and the corresponding $\mathbf{M}$ is chosen such that receptive field is so narrow that it can only be activated by this data point, the corresponding linear model can fit this data point with zero error. The function approximation result would strongly tend towards overfitting. It is this property that has led many learning algorithms to resort to competitive learning with fixed number of local receptive fields: the global competitive process will prevent receptive fields from modeling just one data point [MD88, JJ94]. But allowing for global competition takes away the property of being a local learner, even if the receptive fields are actually spatially localized.

©Sethu Vijayakumar 1998

An alternative way to avoid this overfitting effect is to use leave-one-out cross validation. The cost function to be minimized changes from eq.(8.25) to

$$J''[\mathbf{M}] = \frac{1}{W}\sum_{i=1}^{M} w_i(y_i - \hat{y}_{i,-i})^2, \tag{8.27}$$

where the notation $\hat{y}_{i,-i}$ denotes that the prediction of the i-th data point is calculated from training the learning system with the i-th data point excluded from the training set. Leave-one-out cross validation, however, is usually computationally very expensive since an M-fold training of the learning system is required, for M data points in the training set. Furthermore, for example, in sigmoidal neural networks, it may be unclear how to combine the resultant M different solutions of the parameters to the obtain a final solution. However, for *linear* regression problems, there is an exception rendering these concerns irrelevant. Due to the Sherman-Morrison-Woodbury theorem (refer Bensley, Kuh & Welsh [BKW80]), eq.(8.27) can be re-written as

$$J''[\mathbf{M}] = \frac{1}{W}\sum_{i=1}^{M} w_i(y_i - \hat{y}_{i,-i})^2 = \frac{1}{W}\sum_{i=1}^{M} \frac{w_i(y_i - \hat{y}_i)^2}{(1 - w_i\tilde{\mathbf{x}}_i{}^T\mathbf{P}\tilde{\mathbf{x}}_i)^2}, \tag{8.28}$$

where $\mathbf{P}$ is the inverted covariance matrix defined in eq.(8.21). The modified criterion J'' states that the leave-one-out cross validation error can be obtained without M-fold training of the learning system but rather by a one shot training followed by the adjustment of the weighted mean squared error as shown in eq.(8.28).

Unfortunately, there is still a point of concern with eq.(8.28). Minimizing the locally weighted leave-one-out cross validation results in a *consistent* learning system, i.e. with an increasing number of training points the receptive fields will shrink to a very small size. The advantage of this behavior is that function approximation becomes asymptotically unbiased, i.e., consistent, but as a disadvantage, an ever increasing number of receptive fields will be required to represent the approximated function.

This problem can be avoided by introducing a penalty term in eq.(8.28) and modifying it as

$$J_{DIST}[\mathbf{M}] = \frac{1}{W}\sum_{i=1}^{M} \frac{w_i(y_i - \hat{y}_i)^2}{1 - w_i\tilde{\mathbf{x}}_i{}^T\mathbf{P}\tilde{\mathbf{x}}_i} + \gamma \sum_{n,m=1}^{L} D_{nm}^2, \tag{8.29}$$

where the scalar γ determines the strength of the penalty and the elements of distance metric $\mathbf{D}$ are represented as D_{nm}. Eq.(8.29) represents the final version of the cost function we are going to minimize in order to update the distance metric $\mathbf{D}$. By penalizing the sum of the squared coefficients of the distance metric $\mathbf{D}$, we are essentially penalizing the second derivatives of the function at the site of a receptive field. This is similar to the approaches taken in spline fitting [CW79] and acts like a low pass filter: the higher the penalty term, the more smoothing (and thus, bias) will be introduced locally. Another positive effect of the penalty term is that the

©Sethu Vijayakumar 1998

introduction of the bias reduces the variance of the function estimate, a problem associated with local function fitting methods [Fri84]. For further theoretical analysis on the asymptotic properties of the optimization criterion J_{DIST} (eq.(8.29)), the reader is referred to Section 4 of [SA97].

The update of the distance metric $\mathbf{D}$ is a gradient descent in J_{DIST}. Since, $\mathbf{D} = \mathbf{M}^T\mathbf{M}$, we look at the update for $\mathbf{M}$ to maintain the positive definiteness of the distance metric. Hence, now we will see how to minimize eq.(8.29) incrementally by adjusting $\mathbf{M}$ by gradient descent with a learning rate α:

$$\mathbf{M}^{(n+1)} = \mathbf{M}^{(n)} - \alpha \frac{\partial J_{DIST}}{\partial \mathbf{M}}. \tag{8.30}$$

The derivative in eq.(8.30) can be written as

$$\frac{\partial J_{DIST}}{\partial \mathbf{M}} = \frac{\partial}{\partial \mathbf{M}} \left(\sum_{i=1}^{M} \frac{w_i (y_i - \hat{y}_i)^2}{W(1 - w_i \tilde{\mathbf{x}_i}^T \mathbf{P} \tilde{\mathbf{x}_i})^2} + \gamma \sum_{n,m=1}^{L} D_{nm}^2 \right). \tag{8.31}$$

Let

$$J_{1,i} = \frac{w_i (y_i - \hat{y}_i)^2}{W(1 - w_i \tilde{\mathbf{x}_i}^T \mathbf{P} \tilde{\mathbf{x}_i})^2}, \tag{8.32}$$

$$J_2 = \gamma \sum_{n,m=1}^{L} D_{nm}^2. \tag{8.33}$$

Then, applying chain rule, eq.(8.31) can be written as

$$\frac{\partial J_{DIST}}{\partial \mathbf{M}} = \frac{\partial}{\partial \mathbf{M}} \left(\sum_{i=1}^{M} J_{1,i} + J_2 \right) \tag{8.34}$$

$$= \sum_{i=1}^{M} \sum_{j=1}^{M} \frac{\partial J_{1,i}}{\partial w_j} \frac{\partial w_j}{\partial \mathbf{M}} + \frac{\partial J_2}{\partial \mathbf{M}}. \tag{8.35}$$

Without storing data in incremental learning, we cannot use cross validation and thus, cannot obtain the true gradient in eq.(8.31). The usual approach to derive stochastic gradient would be to drop the two sums in eq.(8.35). However, such a gradient would be quite inaccurate since the first term of eq.(8.31) would always be positive : shrinking the receptive field reduces the weight of a data point and thus, it's contribution to the weighted error. It turns out that we are able to derive a much better stochastic approximation by working on the lines shown in [SA97]. Given one training point $(\mathbf{x}, y)$ and it's associated weight w from eq.(8.2), the derivative for this point can be approximated as

$$\frac{\partial J_{DIST}}{\partial \mathbf{M}} \approx \sum_{i=1}^{M} \frac{\partial J_{1,i}}{\partial w} \frac{\partial w}{\partial \mathbf{M}} + \frac{w}{W} \frac{\partial J_2}{\partial \mathbf{M}} = \frac{\partial w}{\partial \mathbf{M}} \sum_{i=1}^{M} \frac{\partial J_{1,i}}{\partial w} + \frac{w}{W} \frac{\partial J_2}{\partial \mathbf{M}}. \tag{8.36}$$

Summing eq.(8.36) over all data points and recalling that W stands for the sum of all weights (refer eq.(8.26)), eq.(8.36) can be verified to result in eq.(8.35). Despite

©Sethu Vijayakumar 1998

the presence of the term $J_{i,i}$, it is now possible to obtain an incremental version of the stochastic derivative in eq.(8.36) by introducing the following "memory traces". Let us define the cross validation error at the current point $(\mathbf{x}, y)$ as

$$e_{cv} = (y - {\beta^{(n)}}^T \tilde{\mathbf{x}}) \tag{8.37}$$

and a scalar h, using $s_{xx,i}^{(n+1)}$ of eq.(8.24), as

$$h = w \sum_{i=1}^{L+1} \frac{\tilde{x}_i^2}{s_{xx,i}^{(n+1)}}. \tag{8.38}$$

Also, let us define and initalize the scalar cumulative weight trace as $W^{(0)} = 0$, the scalar cumulative cross validation errors as $E^{(0)} = 0$, while the vector trace $\mathbf{H}^{(0)}$ is set to the zero vector and the matrix trace $\mathbf{R}^{(0)}$ to a matrix with all zero elements. Then, we can update the the following "memory traces" by the recursive formula as

$$\begin{aligned}
W^{(n+1)} &= \lambda W^{(n)} + w, && (8.39)\\
E^{(n+1)} &= \lambda E^{(n)} + w e_{cv}^2, && (8.40)\\
\mathbf{H}^{(n+1)} &= \lambda \mathbf{H}^{(n)} + \frac{w e_{cv} \tilde{\mathbf{x}}}{1-h}, && (8.41)\\
\mathbf{R}^{(n+1)} &= \lambda \mathbf{R}^{(n)} + \frac{w^2 e_{cv}^2 \tilde{\mathbf{x}} \tilde{\mathbf{x}}^T}{1-h} && (8.42)
\end{aligned}$$

Let the elements of the matrix $\mathbf{M}$ be represented as M_{ij}. The resultant incremental version of the derivative of eq.(8.36) becomes

$$\frac{\partial J_{DIST}}{\partial \mathbf{M}} \approx \frac{\partial w}{\partial \mathbf{M}} \sum_{i=1}^{M} \frac{\partial J_{1,i}}{\partial w} + \frac{w}{W^{(n+1)}} \frac{\partial J_2}{\partial \mathbf{M}}, \tag{8.43}$$

where

$$\begin{aligned}
\frac{\partial w}{\partial M_{rl}} &= -\frac{1}{2} w (\mathbf{x} - \mathbf{c})^T \frac{\partial \mathbf{D}}{\partial M_{rl}} (\mathbf{x} - \mathbf{c}), && (8.44)\\
\frac{\partial J_2}{\partial M_{rl}} &= 2\gamma \sum_{n,m=1}^{L} D_{nm} \frac{\partial D_{nm}}{\partial M_{rl}}, && (8.45)\\
\frac{\partial D_{nm}}{\partial M_{rl}} &= M_{rm}\delta_{nl} + M_{nr}\delta_{ml}, && (8.46)\\
\sum_{i=1}^{M} \frac{\partial J_{1,i}}{\partial w} &\approx -\frac{E^{(n+1)}}{(W^{(n+1)})^2} + \frac{e_{cv}^2 - 2(y - \tilde{\mathbf{x}}^T \beta^{(n+1)})}{W^{(n+1)}} (\mathbf{p}^{(n+1)}, \mathbf{H}^{(n)}) \\
&\quad - \frac{1}{W^{(n+1)}} ((2\mathbf{p}^{(n+1)} {\mathbf{p}^{(n+1)}}^T), \mathbf{R}^{(n)}) && (8.47)
\end{aligned}$$

where δ_{ml} is the Kroneker's delta, $(.,.)$ represents the Schmidt's inner product of two homomorphic matrices or vectors such that

$$(\mathbf{Q}, \mathbf{V}) = \sum_{i,j} Q_{ij} V_{ij} \tag{8.48}$$

©Sethu Vijayakumar 1998

and $\mathbf{p}^{(n)}$ is a vector whose elements are defined as

$$p_i^{(n)} = \frac{\tilde{x}_i^2}{s_{xx,i}^{(n)}} \qquad 1 \leq i \leq L+1. \tag{8.49}$$

Deriving this derivative is possible due to the fact that an application of the Sherman-Morrison-Woodbury theorem allows us to take derivatives through the inverted covariance matrix $\mathbf{P}$ [BKW80] and that a sum of the form $\sum \mathbf{v}_i^T \mathbf{Q} \mathbf{v}_i$ can be written as $(\mathbf{Q}, \sum \mathbf{v}_i \mathbf{v}_i^T)$. Furthermore, the computation of the derivative has become much simpler because the PCA dimensionality reduction stage decorrelates the input to the regression stage, making it trivial to compute the inversion of the covariance matrices. It is interesting to note that the stochastic derivative of eq.(8.43) is not just concerned with reducing the error of the current training point as in many other learning algorithms but rather takes into account the previously encountered training data too through the memory traces of eq.(8.39)~(8.42). Thus, both β and $\mathbf{M}$ update in LASS are not greedy with respect to the current training sample, a characteristic which will contribute favorably to the speed and robustness of incremental learning.

8.6 The working of the LASS algorithm

The LASS algorithm proceeds as following. The entire system is initialized with no processing element. Every piece of training data $(\mathbf{x}, y)$ is used to update all the existing elements. If no element is activated (refer eq.(8.2)) more than a threshold w_{gen}, a new LASS element is created with its receptive field center $\mathbf{c}$ in eq.(8.2) initialized to $\mathbf{c} = \mathbf{x}$. All other parameters of the new LASS element are initialized to zero, except for $\mathbf{D}$ and $\mathbf{W}^{PCA}$. The distance metric $\mathbf{D}$ is set to a user supplied value $\mathbf{D}_{def}$, while the coefficients W_{ii}^{PCA} are set to "1", and all other coefficients to zero. The initial dimensionality of the LWPCA starts out with $L = 2$, although the regression will only use $L - 1$ inputs. Having one more LWPCA output than used in the regression allows the algorithm to monitor when the dimensionality of the regression stage should be increased, as explained below.

The dimensionality of the regression stage is augmented based on a variance threshold criterion. For this purpose, each LASS unit keeps an incremental record of the variances of its L LWPCA outputs by maintaining a vector $\mathbf{v}_{PCA} \equiv (v_{PCA,i})$ as

$$v_{PCA,L}^{(n+1)} = (\lambda W^{(n)} v_{PCA,L}^{(n)} + w x_{reg,L}^2)/(W^{(n)} + w) \tag{8.50}$$

©Sethu Vijayakumar 1998

We define an appropriate PCA cutoff threshold, θ_{PCA}. If the condition

$$v_{PCA,L} / \sum_{i=1}^{L} v_{PCA,i} > \theta_{PCA} \tag{8.51}$$

is true, the dimensionality L is incremented by one and appropriate coefficients are added in $\mathbf{W}^{PCA}$ and $\mathbf{b}$. The learning rule of eq.(8.11) guarantees that the elements of the variance vector $\mathbf{v}_{PCA}$ are in descending order[San89] such that only the L-th variance has to be monitored. To avoid premature adding of dimensions, it is useful to also monitor the rate of change of the variances and add dimensions only if the rate of change is close to zero.

It is important to note that adding dimensions does not disturb the results obtained by the previously trained LASS parameters. A new dimension in the LWPCA adds a row to $\mathbf{W}^{PCA}$, but the learning rule of eq.(8.11) ensures that updates of coefficients of $\mathbf{W}^{PCA}$ are not affected by coefficients whose row index is larger. Thus, the new row is trained entirely independently. Similarly, adding a coefficient to $\mathbf{b}$ just adds a new element to an additive regression. As shown in eq.(8.22), the regression updates for each dimension are independent of each other due to the decorrelation of $\mathbf{x}_{reg}$ in the LWPCA. Due to ordering of the variances, it can also be argued statistically that regression coefficients with a small variance have a low prior probability of contributing significantly to the regression output, since the confidence in a regression coefficient depends inversely proportional on the variance. All these facts contribute positively to an incremental mechanism with minimal interference.

In sum, LASS is a constructive learning algorithm in two different senses. First, LASS elements are added whenever a training point in input space does not sufficiently activate any existent LASS element. This process will guarantee that the entire input distribution of the training data is quickly covered by LASS. Second, within each LASS element the dimensionality of the regression stage can grow until the LWPCA models a user specified fraction of the local variance of the inputs. The size and shape of the local region of validity of a LASS element, however, is determined by a goodness of fit criterion in regression space, thus coupling the choice of a local subspace to the regression stage, not unlike the classification algorithm of Kambhatla & Leen [KL94]. These features ensures the quality of the regression result. The adjustable parameters in each LASS element are the LWPCA weights $\mathbf{W}^{PCA}$, the regression parameters β and the distance metric $\mathbf{D}$, all of which are trained with second order learning techniques. We will provide the pseudocode for implementation of the LASS algorithm in the next section and show results of performance of the algorithm in empirical evaluations on artificial data, real robot motion data and human arm movement data in Chapter 9.

©Sethu Vijayakumar 1998

8.7 LASS pseudocode implementation

A compact pseudocode implementation of the LASS algorithm is given below. The initializations of the newly created LASS subnets are as mentioned in Section 8.6.

Initialize the LASS network with no subnets (Receptive Fields);

For every new training sample $(\mathbf{x},\mathbf{y})$:

(a) For k=1 to #subnets:

- calculate activation w_k (refer eq.(8.2))
- compute weighted PCA update $\Delta\mathbf{W}^{PCA}$ (refer eq.(8.11))
- compute distance metric update $\Delta\mathbf{D}_k$ (refer eq.(8.43))
- compute new regression β_k (refer eq.(8.22))
- check if another PCA dimension is to be added by comparing variance ratio of last added dimension
  - $\star$ If yes, add and change affected variables

end.

(b) If no subnet is activated more than w_{gen}:

- create a new subnet (RF) with $\mathbf{c} = \mathbf{x}$.

end.

end;

In the next chapter, we will analyze the results of the working of LASS on artificial and real robot and human motion data and use it to approximate the sensorimotor map of a 7-DOF robot arm.

©Sethu Vijayakumar 1998

©Sethu Vijayakumar 1998

9

LASS High Dimensional Learning Performance

9.1 Introduction

In Part III of the dissertation, we have built up the LASS incremental non-parametric learning scheme in stages and provided the theoretical basis behind the working of each of it's components.

In this chapter, we first demonstrate the working of the LASS algorithm through an artificial toy problem which has redundancies embedded in the input dimensions. The effectiveness of a method has to be, of course, measured on real world problems. Hence, we go on to apply this learning scheme in learning of the highly complex and non-linear sensorimotor map of a 7 DOF robot arm. The convergence of regression and the average dimensionality of data employed for approximation is traced and the learned inverse dynamics is used to actually control the robot arm in following a particular trajectory.

Finally, regression results for data collected from a human arm motion is used to learn an idealized model of the internal dynamics of the arm in the sensorimotor loop. Results are compared against the standard neural network based global learners and also against the Receptive Field Weighted Regression (RFWR) [SA96], a successful local learner in low dimensions.

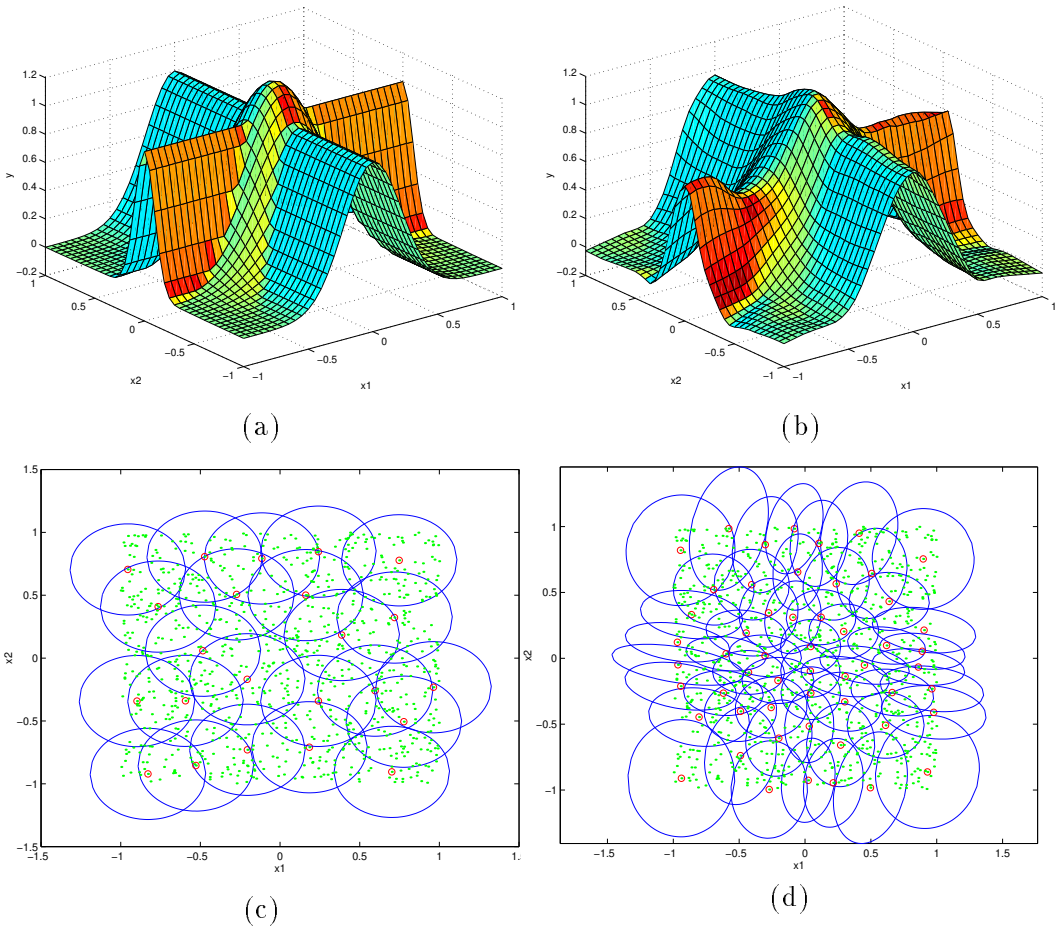


Figure 9.1: (a) Surface plot of function 'cross' to be approximated; (b) LASS approximation results for the 10-dimensional input data set created by projecting the 'cross' data onto 10 dimensions (8 of them redundant) and rotating (c) Receptive field in input space (projected back onto 2-D) after a few epochs of learning, given by contour lines of 0.1 isoactivation along with centers marked by a 'o' (training data displayed by small dots) (d) receptive fields after the completion of training

9.2 Learning in 10-dimensional artificial example

In the first example we will use a synthetic data set that allows to illustrate function fitting results with LASS graphically. The task is to approximate

$$y = \max\{\exp(-10x_1^2), \exp(-50x_2^2, 1.25\exp(-5(x_1^2 + x_2^2)))\} + \mathrm{N}(0, 0.01) \qquad (9.1)$$

from noisy data set of 500 samples, drawn uniformly from the unit square. This function consists of a narrow and a wide ridge which are perpendicular to each other, and a Gaussian bump at the origin as shown in Fig. 9.1(a). The test data set consists of 1681 data points corresponding to the vertices of a 41x41 grid over the unit square; the corresponding output values are the exact function values. The

©Sethu Vijayakumar 1998

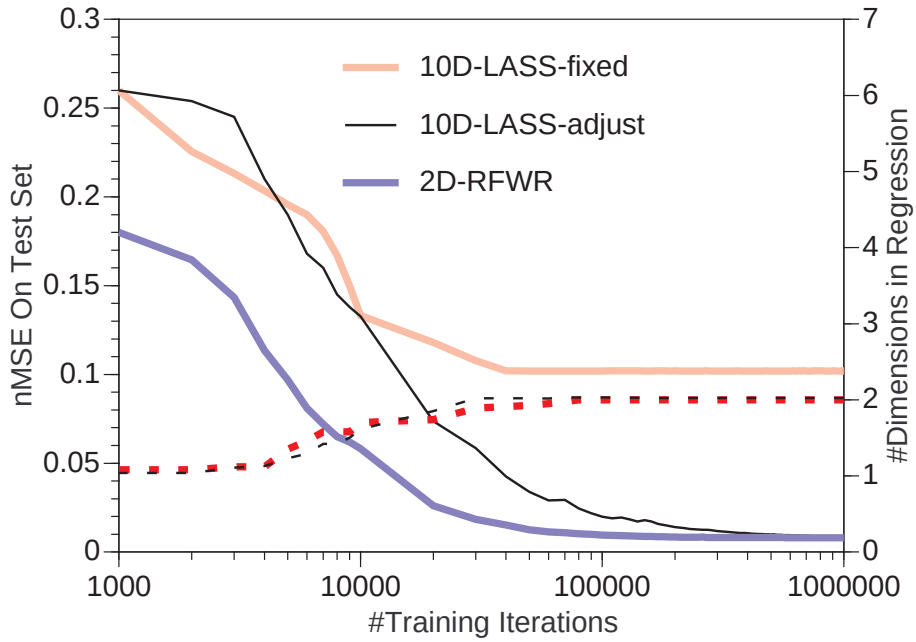


Figure 9.2: $nMSE$(solid lines) and dimensionality of regression(dashed lines) as function of training iterations for 10-D LASS approximation with fixed and adjusted distance metric and a baseline comparison with 2-D RFWR. One training iteration corresponds to one incremental presentation of a training sample.

approximation error is measured as a normalized mean squared error, $nMSE$, i.e, the MSE on the test set normalized by the variance of the outputs of the test set. The initial parameters of LASS are initialized as follows. The default distance metric (see eq.(8.2), refer Section 8.6) is initalized to $\mathbf{D}_{def} = 20\mathbf{I}$($\mathbf{I}$ is the identity matrix), the receptive field generation weight threshold (refer Section 8.6) is set at $w_{gen} = 0.2$, while the PCA cutoff threshold (see eq.(8.51)) was set at $\theta_{PCA} = 0.03$. The PCA learning rate (see eq.(8.11)) was set at $\eta = 1$, the distance metric update rate of eq.(8.30) was $\alpha = 100$ and the forgetting factor (see eq.(8.23)) was set to $\lambda = 0.9995$.

In the test for LASS, we augmented the input space by 8 additional dimensions whose values were zero, and transformed this input space by a 10 dimensional randomly chosen rotation matrix. Thus, the task of LASS was to recover this low dimensional function now embedded in a high dimensional space. As a comparison, LASS was trained with and without the adjustment of the distance metric D. Fig.9.1 shows the learning results visually while Fig.9.2 illustrates the course of learning for both tests. It takes about 50,000 iterations until the LWPCA converges initially. At this point, the average local dimensionality of the regression starts to increase and saturates at 2, which corresponds to the actual dimensionality of the data. Simultaneously, the $nMSE$ drops quickly and for a fixed distance metric, the algorithm converged

©Sethu Vijayakumar 1998

with an $nMSE = 0.1$. On the other hand, when we allowed the adjustment of the distance metric $\mathbf{D}$ based on the regression (refer eq.(8.30)), an $nMSE = 0.01$ was achieved, with the LWPCA again correctly saturating the regression dimensions employed at 2.

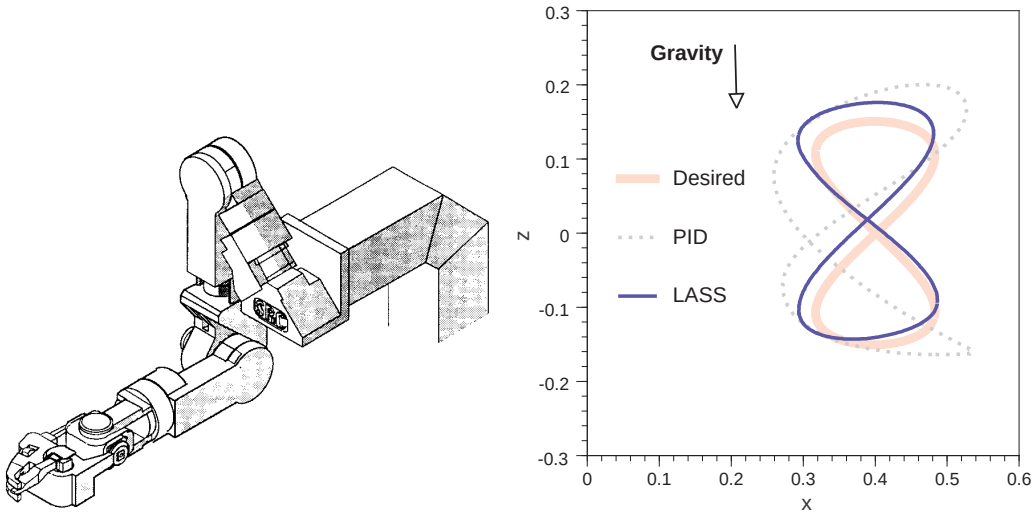
Fig.9.1(b) shows a typical example of the excellent reconstruction of the function after rotating the results back into the original low dimensional space. Furthermore, a visual analysis of the size and shape of the receptive fields confirms that LASS fulfills our expectations. The initially large receptive fields (see Fig.9.1(c)) adjusts it's region of locality based on the function's curvature, hence, shrinking along high frequency directions while stretching over largely linear areas, as shown by the receptive field orientation at the end of learning (see Fig.9.1(d)). The number of receptive fields employed for the approximation increased with the progress of training and saturated at a value of 51 for this example. The $nMSE$ achieved by LASS was indistinguishable to the results achieved in 2-dimensional approximation of the same function by RFWR (see Fig.9.2), a robust local learning technique for low dimensional non-parametric regression [SA96]. It should also be noted that the $nMSE$ starts at a fairly low value after only 1,000 iterations, despite the system only employing one dimensional regressions at this point. The fast approximation speed is due to the second order learning employed by both the LWPCA and LWR stages of LASS. As a baseline comparison, a sigmoidal neural network requires about 100 hidden units and about 20,000,000 iterations to accomplish a similar function fitting result.

9.3 Learning a 21-dimensional sensorimotor map

9.3.1 Sensorimotor map of a 7-DOF anthropomorphic robot arm

In the previous section, we verified the effectiveness of the LASS algorithm on an artificial example. In our second set of experiments, we will try to approximate the inverse dynamics model of a 7 DOF anthropomorphic robot arm (Sarcos Dexterous Arm, see Fig.9.3(a)), a 21 dimensional to 7 dimensional highly nonlinear map using the LASS algorithm. For this purpose, we used a data set consisting of 45,000 data points, collected at 100Hz from the actual robot performing various rhythmic and discrete movement tasks. The data was randomly split into half to obtain a training set and testing set. Fig.9.4 shows the learning results in comparison to a linear regression model. Already after 10,000 iteration, LASS performed as good as the linear regression model despite employing only 1 regression dimension at this stage. After about 100,000 iterations (roughly 4 passes through the training set or

©Sethu Vijayakumar 1998



(a) (b)

Figure 9.3: (a) Sketch of SARCOS Dexterous Arm (b) Trajectory tracking comparison of the SARCOS arm using a PID controller against using a non-parametric inverse dynamics model learned by LASS.

15 minutes of real-time data), LASS already accomplished a $nMSE$ (averaged over all 7 output dimensions) of under 0.05. The system converges at $nMSE = 0.03$ while employing an average of 6 dimensions locally, confirming our hypothesis that physical systems realize locally low dimensional data distributions.

9.3.2 Trajectory tracking performance in a drawing 8 task

An authoritative statement about having successfully learned an inverse dynamics model should not solely be made based on $nMSE$ values alone; the true test is, of course, success in motion synthesis and performance on actual trajectory tracking. To this effect, we used the inverse dynamics model learned by LASS in a trajectory following task of drawing an 'eight' in Cartesian coordinates. In comparison to a low gain, biologically inspired PID controller on the same task, the LASS based controller improved tracking significantly (see Fig.9.3(b)) although a slight constant offset due to incomplete gravity compensation can be noticed.

9.4 Learning with real human arm data

In the final evaluation, we use real movement data collected from human arm movements in random point-to-point reaching and rhythmic scribbling tasks. Kinematic

©Sethu Vijayakumar 1998

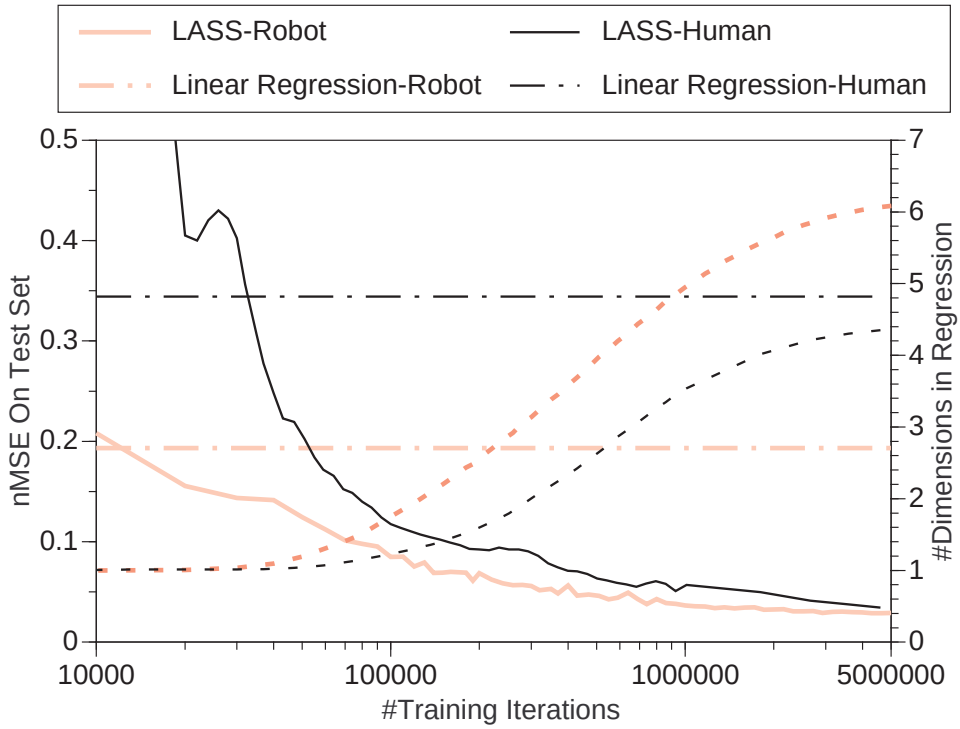


Figure 9.4: Average dimensionality and typical learning curves for LASS in real movement data of humans and robots.

arm data was recorded with optical recording equipment. From this data, the 7 joint positions of the arm and their associated velocities and accelerations were recovered. Under the assumption of a rigid body dynamics model, corresponding joint torques were computed using biologically plausible values for inertia and mass parameters. LASS was trained to fit the inverse dynamics model of this data, a mapping from the joint positions, velocities, and accelerations to the joint torques (a 21 to 7 dimensional function). This test was intended to determine whether a successful nonparametric approximation of the rigid body inverse dynamics, given the measured input distribution from the human subjects, could be obtained in locally reduced subspaces. Results of this evaluation showed that LASS achieved an $nMSE$ of 0.04 on test sets by employing only between 4-5 dimensions on the average for the regression (see Fig.9.4). This strongly supports our assumption that real movement data are *locally* low dimensional. As a baseline comparison, the linear regression result on the human data can do no better than an $nMSE$ of 0.35 on the test data.

©Sethu Vijayakumar 1998

9.5 Comments

Based on the empirical evaluations and tests carried out on real human and robot movement data, it was shown that the LASS algorithm could be used to handle high dimensional problems in a novel way. One of the major goals of this part of the dissertation is to emphasize an important point, i.e, local learning for regression in high dimensional spaces may not be as complicated as previously thought. The rationale for this statement is based on the assumption that data distributions, despite being *globally* high dimensional, are *locally* often of only low dimensional structure. Evaluations of the real movement data provided strong support for this assumption. For a synthetic and an actual human and robot data sets, we illustrated that LASS achieved the expected performance: in all cases, locally low dimensional data distributions were detected and exploited appropriately. In contrast to our previously developed learning methods whose computational complexity is more than quadratic, LASS scales linearly to the number of input dimensions.

However, one major drawback of these kinds of data driven local approaches is that the results are strongly based on the assumption that we have enough data support in the region of input space where we want to learn. One has to look into problems of exploration and dynamic programming if this kind of data support is not automatically guaranteed. However, at least in most control tasks, the robot usually moves (and hence, provides data) in the broad region of interest of the function space.

An open point of research in LASS concerns how the regression analysis could influence the LWPCA. At the current stage of LASS, LWPCA proceeds independently of LWR, which, from a statistical point of view, is not satisfying as the quality of the regression depends on the distribution of the input data [SA97]. Empirical evaluations will provide insight into how much this shortcoming affects the quality of learning. Performing LWPCA in joint data space, or employing alternative techniques, like partial least squares regression, have shown promising results in some pilot studies.

©Sethu Vijayakumar 1998

©Sethu Vijayakumar 1998

Part IV

Conclusion

10

Conclusions & Future Objectives

In this chapter, we summarize the major contributions of this dissertation and list out the results that have been derived. A compilation of future research directions and some open problems remaining in direct association with the work presented here is also detailed out.

10.1 Conclusions

The major contributions of this thesis are compiled below.

We have reviewed the functional analytic approach to learning in neural networks and laid out the framework for learning based on an analysis at the level of function spaces in Chapter 2. Various optimization criterion for learning were considered and importance of using a criterion which reduces error in the original function space rather than the sampled space was emphasized in accordance with the work of Ogawa et al.[Oga90, Oga92b, OO86].

We have discussed the problems of model selection and the bias variance dilemma which comes up during the selection of the function search space in Chapter 3 and have shown that this framework is well suited for use with standard model selection strategies.

The definition and need for *exact* incremental learning is provided in Chapter 4.

Methods for incrementally computing the Wiener learning operator and the learned function using only the newly available training data and the results of the learning carried out so far are provided. The general results for other optimization criterion, done as an extension of this work [Nak97], is also provided here. A concrete pseudocode for implementation of the incremental learning algorithm and simulation results proving the effectiveness of the algorithm are also shown. The important point to be noted is that the incremental learning results are *exact* in the sense that they exactly coincide with the results of batch learning on the entire data set.

An example of using this framework for learning in real world problems (here, the sensorimotor map of a 2 DOF robot arm) by incorporating apriori knowledge effectively is shown in Chapter 5.

During the course of the dissertation, we have emphasized the importance and need of not just using the given training data but dynamically selecting the training data with a view to improving generalization ability. Chapter 6 provides a method of selecting the optimal training data, referred to as the *active data selection*, for the Wiener optimization criterion.

Techniques described under exact incremental learning work surprisingly well under the presence of strong apriori knowledge and are particularly useful in problems where the dimensionality is not too high. But when we consider learning in high dimensional spaces without strong apriori knowledge, this and most other parametric as well as non-parametric methods break down due to the complexity. However, we found evidence that in problems of motor control, the data generated by biological and artificial movements systems, though being high dimensional and sparse globally, usually are distributed densely on a low dimensional hyperplane *locally*. In order to exploit this, Chapter 7 compared different techniques for local dimensionality reduction based on Monte Carlo evaluation and found a suitable candidate in Locally Weighted PCA (LWPCA).

The LWPCA dimensionality reduction is used on top of a locally weighted regression to implement the Local Adaptive Subspace Regression (LASS). LASS is an incremental learning algorithm that uses no apriori knowledge, is completely incremental in allocating resources as well as in incorporating new data and avoids competition between local modules to prevent negative interference. LASS was tested in Chapter 9 and for artificial, robot and human motion data, it was shown to effectively detect and exploit low dimensional local manifolds from high dimensional data and achieve excellent learning results.

To summarize, in the presence of strong apriori knowledge and for optimal generalization, we recommend using the exact incremental learning technique developed in Part I of the dissertation. Active data selection techniques can be used to obtain optimal generalization with minimal training data. However, in the absence of

©Sethu Vijayakumar 1998

apriori knowledge and especially, for learning in high dimensional spaces, LASS is a very effective alternative for obtaining good approximations to the real solution.

10.2 Future Objectives

Some of the future research directions and interesting topics that could be pursued are listed under.

- The exact incremental framework which has been developed in Part I of this dissertation deals with learning only in noiseless cases. Research on extending the results for the *noisy* case is underway in our group.
- The active learning results, i.e., method of selecting optimal training data has been provided for the Wiener criterion. An approach on the similar lines can be used for deriving active learning results for other criterion.
- The LASS algorithm, at present, does the dimensionality reduction of the input data independent of the output values. From a statistical point of view, this is not a very satisfying solution although it may work very well in practice. Future research will look at coupling the dimensionality reduction stage and the regression stage by using techniques like PCA in joint data space.
- The links between a very effectively used classification technique, the Support Vector Machines (SVMs) [Vap96] and the functional analysis based techniques for regression will be pursued.
- One of the major challenges for effectively using the Reproducing Kernel Hilbert Space (RKHS) based methods remains to be construction of suitable search spaces for particular problems or application. This work provides an example for construction of these spaces for the robot control problem. Future work will be directed towards more general methods of construction for a wider class of problems.

©Sethu Vijayakumar 1998

©Sethu Vijayakumar 1998

Acknowledgments

I am indebted to Professor Hidemitsu Ogawa, who took me under his supervision in 1992, for accepting me in his laboratory, for providing an excellent working environment and above all, for his valued guidance and encouragement despite his extremely busy schedule. I would especially like to thank him for being so supportive of my initiatives and putting up with my numerous requests for pursuing joint research projects outside of the university.

I wish to express my gratitude to Dr. Stefan Schaal of the University of Southern California for having encouraged me to take up the joint initiative with ATR Human Information Processing Research Labs, Kyoto and for his guidance and support during the course of the work done there. Special thanks are due to Dr. Mitsuo Kawato for supporting me and making the joint research work at ATR possible.

I am grateful to Dr. Itsuo Kumazawa and Dr. Yukihiko Yamashita for the valuable discussions. I wish to thank Dr. Chris Atkeson and Dr. Kenji Doya for making my short stays at ATR extremely fruitful and scientifically invigorating. Special thanks are due to Mr. Hirabayashi Akira, Ms. Kyoko Satoh and Ms. Mieko Namba for their permanent kindness and help.

I would also like to extend a special 'thanks' to all the students of Ogawa laboratory, the members of the Human Information Processing Labs (ATR) and the ERATO Kawato Dynamic Brain Project group for the wonderful time, academically as well as in sharing those lighter moments, which went a long way in motivating this work.

I would like to express my gratitude to Mombusho, the Ministry of Education, Science, Sports and Culture, Government of Japan, for enabling me to come to Japan and providing the necessary financial support for this research. Last but certainly not the least, I would like to thank my parents and sister, who have been a constant source of inspiration throughout my life and have egged me on whenever I faltered.

©Sethu Vijayakumar 1998

Bibliography

[AH77] H.C. Andrews and B.R. Hunt. *Digital Image Restoration.* Prentice-Hall, New Jersey, 1977.

[Alb72] A. Albert. *Regression and the Moore-Penrose Pseudoinverse*, Vol. 94 of *Mathematics in Science and Engineering.* Academic Press, 1972.

[AMS97] C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, Vol. 11, No. 4, pp. 76–113, 1997.

[Ber86] M. Bertero. Regularization methods for linear inverse problems. In C.G.Talenti, editor, *Inverse Problems*, Berlin, 1986. Springer Verlag.

[Bis95] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[BKW80] D.A. Belsley, E. Kuh, and R.E. Welsch. *Regression diagnostics: Identifying influential data and sources of collinearity.* Wiley, New York, 1980.

[Coh94] D.A. Cohn. Neural network exploration using optimal experiment design. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, Vol. 6, pp. 679–686. Morgan-Kaufmann, San Mateo, 1994.

[CP87] P. Carnevali and S. Patarnello. Exhaustive thermodynamic analysis of Boolean learning networks. *Europhysics Letters*, Vol. 4, pp. 1199–1204, 1987.

[Cri89] F. Crick. The recent excitement about neural networks. *Nature*, Vol. 337, pp. 129–132, 1989.

[CW79] P. Craven and G. Wahba. Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross validation. *Numerical Mathematics*, Vol. 31, pp. 377–403, 1979.

[DD95] J. Daugman and C. Downing. Gabor wavelets for statistical pattern recognition. In M.A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pp. 414–420. MIT Press, Cambridge, MA, 1995.

[deF90] R.J. deFigueiredo. A new nonlinear functional analytic framework for modelling artificial neural networks. In *Proceedings, International Symposium on Circuits and Systems (New Orleans)*, pp. 723–726, 1990.

[DO96] G. Deco and D. Obradovic. *An information theoretic approach to neural computing*. Perspectives in Neural Computing. Springer-Verlag, 1996.

[DSTK97] N. Dunkin, J. Shawe-Taylor, and P. Koiran. A new incremental learning technique. In *Proceedings of the 8th Italian Workshop on Neural Networks, WIRN Vietri-96*, pp. 112–118. Springer, 1997.

[Efr79] B. Efron. Computers and theory of statistics. *SIAM Review*, Vol. 21, pp. 460–480, 1979.

[Eve84] B.S. Everitt. *An introduction to latent variable models*. Chapman and Hall, London, 1984.

[Fed72] V.V. Federov. *Theory of optimal experiments*. Academic Press, New York, 1972.

[FF93] I.E. Frank and J.H. Friedman. A statistical view of some chemometric regression tools. *Technometrics*, Vol. 35, No. 2, pp. 109–135, 1993.

[FK87] A. Fernald and P. Kuhl. Acoustic determinants of infant preferences for Motherese speech. *Infant Behaviour and Development*, Vol. 10, pp. 279–293, 1987.

[Fri84] J.H. Friedman. A variable span smoother. Technical Report 5, Department of Statistics, Stanford University, 1984.

[Fuk96] K. Fukumizu. Active learning in multilayer perceptrons. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, Vol. 8, pp. 295–301. The MIT Press, Cambridge, MA, 1996.

[GBD92] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias-variance dilemma. *Neural Computation*, Vol. 4, No. 4, pp. 1–58, 1992.

©Sethu Vijayakumar 1998

[Gre51] U. Grenander. On empirical spectral analysis of stochastic processes. *Ark. Matemat.*, Vol. 1, pp. 503–531, 1951.

[HJ94] R.A. Horn and C.R. Johnson. *Matrix analysis.* Press Syndicate of the University of Cambridge, 1994.

[HT90] T.J. Hastie and R.J. Tibshirani. *Generalized additive models.* Chapman and Hall, London, 1990.

[JC95] C. Jutten and R. Chentouf. A new scheme for incremental learning. *Neural Processing Letters*, Vol. 2, No. 1, pp. 1–4, 1995.

[JJ94] M.I. Jordan and R. Jacobs. Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, Vol. 6, No. 2, pp. 181–214, 1994.

[KH92] A. Krogh and J.A. Hertz. A simple weight decay can improve generalization. In J.E. Moody, S.J. Hanson, and R.P. Lippman, editors, *Advances in Neural Information Processing Systems*, Vol. 4, pp. 950–957. Morgan-Kaufmann, 1992.

[KL94] N. Kambhatla and T.K. Leen. Fast non-linear dimension reduction. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, Vol. 6, pp. 152–159. Morgan Kaufmann Publishers, Inc., 1994.

[KN93] V. Kadirkamanathan and M. Niranjan. A function estimation approach to sequential learning with neural networks. *Neural Computation*, Vol. 5, No. 6, pp. 954–975, 1993.

[LS86] L. Ljung and T. Soederstroem. *Theory and practice of recursive identification.* MIT Press, Cambridge, 1986.

[Lut85] S.P. Luttrell. The use of transinformation in the design of data sampling schemes for inverse problems. *Inverse Problems*, Vol. 1, No. 1, pp. 199–218, 1985.

[Mac92] D. Mackay. Information-based objective functions for active data selection. *Neural Computation*, Vol. 4, No. 4, pp. 590–604, 1992.

[Mar77] D. Marr. Artificial intelligence - a personal view. *Artificial Intelligence*, Vol. 9, pp. 37–48, 1977.

[Mas65] W.F. Massy. Principal component regression in exploratory statistical research. *Journal of the American Statistical Association*, Vol. 60, pp. 234–246, 1965.

©Sethu Vijayakumar 1998

[MD88] J. Moody and C. Darken. Learning with localized receptive fields. In D. Toretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings, 1988 Connectionist Summer School*, pp. 133–143. Morgan Kaufmann, San Mateo, CA, 1988.

[Moo92] J.E. Moody. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In John E. Moody, Steve J. Hanson, and Richard P. Lippmann, editors, *Advances in Neural Information Processing Systems*, Vol. 4, pp. 847–854. Morgan Kaufmann Publishers, Inc., 1992.

[Nak97] S. Nakazawa. A study on incremental learning in optimally generalizing neural networks. Master's thesis, Tokyo Institute of Technology, 1997.

[NT89] K.S. Narendra and M.A.L. Thathachar. *Learning Automata: An Introduction*. Prentice-Hall, Englewood Cliffs, N.J., 1989.

[OF96] B.A. Olhausen and D.J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, Vol. 381, pp. 607–609, 1996.

[Oga81] H. Ogawa. Fundamental studies of digital image restoration. *Transactions of IEICE Japan*, Vol. J64-D, No. 7, pp. 570–576, 1981. In Japanese.

[Oga86] H. Ogawa. A unified approach to generalised sampling theorems. In *Proceedings, IEEE-IEICE-ASJ International Joint Conference on ASSP*, pp. 1657–1660. IEEE-IEICE-ASJ, 1986.

[Oga90] H. Ogawa. Neural network theory as an inverse problem. *Journal of IEICE, Japan*, Vol. 73, No. 7, pp. 690–695, 1990. In Japanese.

[Oga92a] H. Ogawa. Karhunen-Loéve subspace. In *Proceedings, 11th International Conference on Pattern Recognition ICPR (The Netherlands)*, Vol. 11, pp. 75–78, 1992.

[Oga92b] H. Ogawa. Neural network learning, generalization and over learning. In *Proceedings, ICIIPS'92 (Beijing)*, Vol. 2, pp. 1–6, 1992.

[Oga95] H. Ogawa. Neural network and generalization ability. *Technical Report of IEICE Japan*, Vol. NC95, No. 8, pp. 57–64, 1995. In Japanese.

[Oja82] E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, Vol. 15, pp. 267–273, 1982.

[OKC95] J. Oh, C. Kwong, and S. Cho, editors. *Neural Networks: The Statistical Mechanics Perspective; CTP-PBSRI Joint Workshop on Theoretical Physics*. World Scientific, 1995.

©Sethu Vijayakumar 1998

[ON84] H. Ogawa and N. Nakamura. Projection filter restoration of degraded images. In *Proceedings, IEEE International Conference on Pattern Recognition (Montreal)*, pp. 601–603. IEEE, 1984.

[OO86] H. Ogawa and E. Oja. Projection filter, Wiener filter and Karhunen-Loève subspaces in digital image restoration. *Journal of Mathematical Analysis and Applications*, Vol. 114, No. 1, pp. 37–51, 1986.

[OOW92] E. Oja, H. Ogawa, and J. Wangviwattana. Principal component analysis by homogeneous neural networks, part i: the weighted subspace criterion. *IEICE Transactions on Information and Systems*, Vol. E75-D, No. 3, pp. 366–375, 1992.

[PG90] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1481–1497, 1990.

[PW93] M. Plutowski and H. White. Selecting concise training sets from clean data. *IEEE Transactions on Neural Networks*, Vol. 4, No. 2, pp. 305–318, 1993.

[Ris86] J. Rissanen. Stochastic complexity and modelling. *Annals of Statistics*, Vol. 14, pp. 1080–1100, 1986.

[Ris89] J. Rissanen. *Stochastic complexity in statistical enquiry*. World Scientific, Singapore, 1989.

[RT82] D.B. Rubin and D.T. Thayer. EM algorithms for ML factor analysis. *Psychometrika*, Vol. 47, No. 1, pp. 69–76, 1982.

[SA96] S. Schaal and C. Atkeson. From isolation to cooperation: An alternate view of a system of experts. In D.S. Touretsky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, Vol. 8, pp. 605–611. MIT press, Cambridge, MA, 1996.

[SA97] S. Schaal and C. Atkeson. Receptive field weighted regression. Technical Report TR-H-209, ATR Human Information Processing Laboratories, Kyoto, Japan, 1997.

[Sai97] S. Saitoh. *Integral Transforms, Reproducing Kernels and their Applications*. Pitman Research Notes in Mathematics No. 369. Longman Publishers, 1997.

[San89] T.D. Sanger. Optimal unsupervised learning in a single layer feedforward neural network. *Neural Networks*, Vol. 2, pp. 459–473, 1989.

©Sethu Vijayakumar 1998

[Sch97] D. Schuurmans. A new metric based approach to model selection. In *Proceedings, Fourteenth National Conference on Artificial Intelligence*. AAAI, 1997. In press.

[Sco92] D.W. Scott. *Multivariate Density Estimation*. Wiley, New York, 1992.

[SP63] K. Steinbuch and U.A.W. Piske. Learning matrices and their applications. *IEEE Transaction on Electronic Comput.*, Vol. 12, pp. 846–862, Dec 1963.

[SS95] P. Sollich and D. Saad. Learning from queries for maximum information gain in imperfectly learnable problems. In G.Tesauro, D.Touretzky, and T.Leen, editors, *Advances in Neural Information Processing System*, Vol. 7, pp. 287–294. The MIT Press, Cambridge, MA, 1995.

[SUF97] D. Schuurmans, L. Ungar, and D. Foster. Charecterizing the generalization performance of model selection strategies. In *Proceedings, Fourteenth International Conference on Machine Learning (ICML-97)*, July 1997. In press.

[Sut92] R.S. Sutton. Adapting bias by gradient descent: An incremental version of Delta-Bar-Delta. In *Proceedings, Tenth National Conference on Artificial Intelligence*, pp. 171–176, 1992.

[SVA97] S. Schaal, S. Vijayakumar, and C. Atkeson. Local dimensionality reduction. In *Advances in Neural Information Processing Systems 10*, Denver, Colorado, 1997. In press.

[Val84] L.G. Valiant. A theory of learnable. *Communications of the ACM*, Vol. 27, No. 11, pp. 1134–1142, November 1984.

[Vap96] V. Vapnik. *The Nature of Statistical Learning*. Springer-Verlag, New York, 1996.

[Vij95] S. Vijayakumar. An incremental approach to training data selection in neural networks. Master's thesis, Tokyo Institute of Technology, 1995.

[VO95] S. Vijayakumar and H. Ogawa. A functional analytic approach to incremental learning in optimally generalizing neural networks. In *Proceedings, IEEE International Conference on Neural Networks ICNN'95 (Perth)*, pp. 777–782, Perth, Australia, 1995. IEEE.

[VS97] S. Vijayakumar and S. Schaal. Local Adaptive Subspace Regression. *Neural Processing Letters*, 1997. In press.

©Sethu Vijayakumar 1998

[WHG90] M. Wann, T. Hediger, and N.N. Greenbaun. The influence of training sets on generalization in feedforward neural networks. In *Proceedings, International Joint Conference on Neural Networks(San Diego)*, Vol. 3, pp. 137–142, 1990.

[Whi89] H. White. Learning in artificial neural networks: A statistical perspective. *Neural Computation*, Vol. 1, No. 4, pp. 425–464, 1989.

[Wil95] P.M. Williams. Bayesian regularization and pruning using a Laplacian prior. *Neural Computation*, Vol. 7, No. 1, pp. 117–143, 1995.

[WK91] S.M. Weiss and C.A. Kulikowski. *Computer Systems That Learn*. Morgan Kaufmann, San Mateo, CA, 1991.

[XJH95] L. Xu, M.I. Jordan, and G.E. Hinton. An alternative model for mixtures of experts. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, Vol. 7, pp. 633–640. The MIT Press, 1995.

[YO92] Y. Yamashita and H. Ogawa. Mutual relations among optimum image restoration filters. *Transactions of IEICE Japan*, Vol. J75-D-II, No. 5, pp. 890–898, 1992. In Japanese.

[YSS97] L. Yingwei, N. Sundararajan, and P. Saratchandran. A sequential learning scheme for function approximation using minimum radial basis function neural networks. *Neural Computation*, Vol. 9, No. 9, pp. 461–478, 1997.

[Zha94] B.T. Zhang. An incremental algorithm that optimizes network size and sample size in one trial. In *Proceedings of the International Conference on Neural Networks ICNN'94*, pp. 215–220, Orlando, Florida, 1994. IEEE.

©Sethu Vijayakumar 1998