Spring 2014

Markov Chain Monte Carlo Bayesian Predictive Framework for Artificial Neural Network Committee Modeling and Simulation

Michael S. Goodrich
Old Dominion University

MARKOV CHAIN MONTE CARLO BAYESIAN PREDICTIVE FRAMEWORK FOR ARTIFICIAL NEURAL NETWORK COMMITTEE MODELING AND SIMULATION

by

Michael. S. Goodrich
B.S. May 1979, Old Dominion University
M.S. May 1992, Old Dominion University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfilment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

MODELING AND SIMULATION

OLD DOMINION UNIVERSITY
March 2014

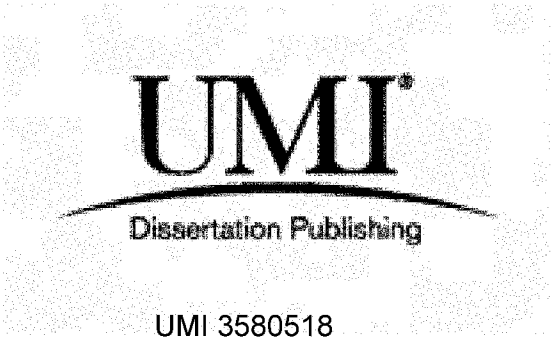
Approved by:

N. Rao Chaganty (Director)

Frederic D. McKenzie (Member)

Jiang Li (Member)

ManWo Ng (Member)



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

ABSTRACT

MARKOV CHAIN MONTE CARLO BAYESIAN PREDICTIVE FRAMEWORK FOR ARTIFICIAL NEURAL NETWORK COMMITTEE MODELING AND SIMULATION

Michael. S. Goodrich
Old Dominion University, 2014
Director: N. Rao Chaganty

A logical inference method of properly weighting the outputs of an Artificial Neural Network Committee for predictive purposes using Markov Chain Monte Carlo simulation and Bayesian probability is proposed and demonstrated on machine learning data for non-linear regression, binary classification, and 1-of-k classification. Both deterministic and stochastic models are constructed to model the properties of the data. Prediction strategies are compared based on formal Bayesian predictive distribution modeling of the network committee output data and a stochastic estimation method based on the subtraction of determinism from the given data to achieve a stochastic residual using cross validation. Performance for Bayesian predictive distributions is evaluated using Bayesian methods, while performance for the residual based method is evaluated using conventional statistical techniques.

ACKNOWLEDGEMENTS

There are many whose guidance and encouragement contributed greatly to the completion of this work. Firstly, I would like to thank my committee members for their patience, guidance, and council. Secondly, I would like to thank my many friends and colleagues including especially Dr. William E. Warner, Ph.D.; Dr. Christine Bacon, Ph.D.; Dr. Eugene A. Stoudenmire, Ph.D.; Dr. Michael A. White, Ph.D.; and Dr. Paul Harvey, Au. D. for their continued exhortations to stay the course. Finally, and most importantly I would like to thank my beloved wife Rochelle for her patience, steadfast encouragement, and long suffering over the many years and hours that she put aside earnest desires that might otherwise have been fulfilled.

TABLE OF CONTENTS

Page

LIST OF TABLES vii

LIST OF FIGURES x

Chapter

1. OVERVIEW 1
   - 1.1 INTRODUCTION 1
   - 1.2 THESIS 2
   - 1.3 RESEARCH OVERVIEW 3
   - 1.4 PROPOSED SOLUTION 7
   - 1.5 RESEARCH OBJECTIVES 8
   - 1.6 ANTICIPATED CONTRIBUTIONS 9
   - 1.7 DOCUMENT OUTLINE 9

2. THEORETICAL BACKGROUND 10
   - 2.1 NEURODES AND PERCEPTRONS 10
   - 2.2 MULTILAYER NETWORKS 13
   - 2.3 THREE LAYER PERCEPTRONS 15
   - 2.4 BAYESIAN PROBABILITY 16

3. RESEARCH REVIEW 20
   - 3.1 MATHEMATICAL BACKPROPAGATION OPTIMIZATION 20
   - 3.2 BAYESIAN BACKPROPAGATION 22
   - 3.3 EVIDENCE FRAMEWORK 22
   - 3.4 AUTOMATIC RELEVANCE DETERMINATION FRAMEWORK 24
   - 3.5 EMPIRICAL BAYES FOR INFINITE NEURAL NETWORKS 24
   - 3.6 EXTENDED ARD FRAMEWORK 25
   - 3.7 OTHER RESEARCH 25
   - 3.8 ISSUES WITH CURRENT APPROACHES 26
   - 3.9 ISSUES NOT ADEQUATELY ADDRESSED 28

4. MARKOV CHAIN MONTE CARLO BAYESIAN PREDICTIVE FRAMEWORK FOR ARTIFICIAL NEURAL NETWORK COMMITTEE MODELING AND SIMULATION 31
   - 4.1 NON LINEAR REGRESSION MODELING 31
   - 4.2 PREDICTIVE MODELING 41
   - 4.3 GENERAL DETERMINISTIC TLP SIMULATION 47
   - 4.4 CONSTRUCTION 50

Chapter Page

5. EVALUATIVE METHODOLOGY AND EXPERIMENTAL DESIGN 51
5.1 OVERVIEW 52
5.2 CROSS VALIDATION 53
5.3 CROSS VALIDATION MEASUREMENTS 56
5.4 K-FACTORIAL DESIGN 61
5.5 DATA 71

6. EXPERIMENTAL CASE STUDY RESULTS AND ANALYSIS 72
6.1 NON-LINEAR REGRESSION CASE STUDY I 72
6.2 CASE STUDY II NON-LINEAR REGRESSION 93
6.3 CASE STUDY III CLASSIFICATION 106
6.4 CASE STUDY IV CLASSIFICATION 111

7. CONCLUSIONS 116

8. FUTURE RESEARCH 118
8.1 REDUCED PARAMETER SPACE FORMULATION 118
8.2 GENERALIZED NORMAL BASED LIKELIHOOD FUNCTION 119
8.3 MCMC METROPOLIS HASTINGS SAMPLING 119
8.4 CROSS ENTROPY EVALUATION 119
8.5 MINIMUM CROSS ENTROPY MOMENT PRIOR 128

BIBLIOGRAPHY 129

VITA 133

LIST OF TABLES

Table		Page
1.	2^3 NMF Factorial BPFANN Design Point Settings	76
2.	2^2 NMF Factorial Matlab Design Point Settings	76
3.	2^3 Factorial BPFANN Training Cross Validation Findings	80
4.	2^3 Factorial BPFANN Predictive Cross Validation Findings	80
5.	2^2 Factorial Matlab Training Cross Validation Findings	89
6.	2^2 Factorial Matlab Predictive Cross Validation Findings	89
7.	2^3 Factorial Bayesian MCMC Design Point Settings.	97
8.	2^2 Factorial Matlab Design Point Settings.	98
9.	2^2 Iris BPFANN Design Point Settings	109
10.	2^2 Iris Factorial Matlab Design Point Settings	110
11.	2^2 WDBC BPFANN Design Point Settings	114
12.	2^2 WDBC Factorial Matlab Design Point Settings	114

LIST OF FIGURES

Figure		Page
1.	Biological Neuron.	10
2.	McCullough-Pitts TLU.	11
3.	Rosenblatt Perceptron.	12
4.	Example Complex Non-Linear Classification Boundaries.	14
5.	Example Three Layer Feed Forward Perceptron Network.	15
6.	Inverse Tangent.	32
7.	Jeffreys vs. Gaussian.	37
8.	Jeffreys vs. Gamma(2,1).	38
9.	2^3 EDCM Measures K-Factorial Matrix.	63
10.	2^3 EDCM ACC Measures K-Factorial Matrix.	63
11.	2^3 SRM K-Factorial Matrix.	64
12.	2^3 BPD K-Factorial Matrix.	65
13.	2^2 EDCM K-Factorial Matrix.	67
14.	2^2 SRM K-Factorial Matrix.	68
15.	2^2 ACC K-Factorial Matrix.	68
16.	2^2 BPFANN Classification K-Factorial Matrix.	70
17.	2^2 Matlab Classification K-Factorial Matrix.	71
18.	Noisy Math Function.	73
19.	Noisy Math Function EDCM 2^3 Cross Validation Results.	75
20.	Noisy Math Function ACC 2^3 Cross Validation Results.	76
21.	Noisy Math Function SRM 2^3 Cross Validation Results.	77
22.	Noisy Math Function BPD 2^3 Cross Validation Results.	77

Figure Page

23. Effects of Bayesian Prior on Regularization. 81

24. 90% BCI Performance by Bayesian Prior. 83

25. 90% BCI Performance by Bayesian Prior. 84

26. σ_{bl} MCMC Sampling by Bayesian Prior. 84

27. σ_{bl} MCMC Sampling by Bayesian Prior. 85

28. Noisy Math Function Matlab EDCM 2^2 Cross Validation Results. 86

29. Noisy Math Function Matlab SRM 2^2 Cross Validation Results. 86

30. Noisy Math Function BPD/SRM (-,-,-) Ground Truth Comparison. 88

31. Noisy Math Function BPD/SRM (-,+,+) Ground Truth Comparison. 89

32. Noisy Math Function Matlab SRM (-,-) Ground Truth Comparison. 89

33. Noisy Math Function Matlab SRM (+-) Ground Truth Comparison. 90

34. Noisy Math Function Matlab BPFANN Residual Comparison. 91

35. Noisy Math Function Matlab BPFANN Residual Comparison. 92

36. Noisy Math Function BPFANN σ_{bl} Sampling. 93

37. Concrete Slump. 95

38. Concrete Slump Flow EDCM 2^3 Cross Validation Results. 96

39. Concrete Slump Flow ACC 2^3 Cross Validation Results. 97

40. Concrete Slump Flow SRM 2^3 Cross Validation Results. 98

41. Concrete Slump Flow BPD 2^3 Cross Validation Results. 98

42. Concrete Slump Flow Matlab EDCM 2^2 Cross Validation Results. 100

43. Concrete Slump Flow Matlab SRM 2^2 Cross Validation Results. 100

44. Concrete Slump Flow BPD/SRM (-,-,-). 102

45. Concrete Slump Flow BPD/SRM (+,+,+). 102

Figure Page

46. Concrete Slump Flow Matlab SRM (-,-). 103

47. Concrete Slump Flow Matlab SRM (+,+). 104

48. Concrete Slump Flow Matlab BPFANN Best Performer Residuals. 104

49. Concrete Slump Flow Matlab BPFANN Worst Performer Residuals. 105

50. Concrete Slump Flow BPFANN σ_{bl} Sampling. 106

51. Iris BPFANN Classification Cross Validation Results. 108

52. Iris Matlab Classification Cross Validation Results. 109

53. Iris Misclassification by both Matlab and BPFANN. 110

54. WDBC BPFANN Classification Cross Validation Results. 113

55. WDBC Matlab Classification Cross Validation Results. 114

56. WDBC Misclassification by both Matlab and BPFANN. 115

CHAPTER 1

OVERVIEW

1.1 INTRODUCTION

Historically, the modeling of Artificial Neural Networks (ANNs) for the simulation of stated input data to stated output data has been framed as a mathematical optimization problem, and for this reason, solutions have been pursued largely in the form of mathematical optimization frameworks. This basic methodology depends upon a calculus based analysis of a suitable mathematical model of the network's response to changes in parameters primarily through the first several orders of derivatives of embedded non-linear functions and is often enhanced with other kinds of optimization techniques such as simulated annealing, genetic algorithms, etc., which are designed to avoid giving a solution which is not the global best.[1]

As the complexity of the network increases and/or as more hidden layers are added for deeper learning, the mathematical forms of the optimization solution become rapidly problematic, for example greatly increasing the explicit coupling between network parameters and complicating the partial derivatives of the network response with respect to the ever more deeply embedded or entangled network parameters and their derivatives. Further, this kind of "tuning" is point solution oriented, raises deep questions about the proper objective of any such optimization, produces

[1]IEEE Transactions and Journals style is used in this thesis for formatting figures, tables, and references.

randomized solutions from arbitrary starting states, and has accumulated a list of disturbing idiosyncrasies[30] calling into question the propriety of this approach.

Other researchers have framed the issue with a somewhat less mathematical approach which does not use an explicit calculus based analysis of a mathematical model of the network behavior, preferring instead Maximum Likelihood (ML) or Maximum A Posteriori (MAP) chiefly in the form of an Empirical Bayesian [2] based methodology, or other optimization based frameworks modified with some selected Bayesian enhancements. These approaches, however, are still optimization based in spirit and suffer their own notable difficulties traceable to what might be termed greedy optimizations instead of more general information as probability discovery.

Improvements in ANN modeling and simulation should be of special interest to the Modeling and Simulation community in light of ANNs' ubiquitous and generic representational capabilities for continuous functions[11][23].

1.2 THESIS

It is our contention that the evolution of frameworks to model ANNs, since their inception approximately thirty years ago, have not adequately explored the modeling of ANNCs from an inferential logic perspective without reference to point optimization. This gives rise to the following

Thesis Statement: *Modeling of data with Artificial Network Committees using Bayesian Probabilistic Inferential methods as opposed to traditional mathematical point optimization methods provides for effective modeling of both deterministic and stochastic components of the data for predictive purposes as measured by the ability to model predictive outputs.*

The purpose of this study is to demonstrate the feasibility of basing ANNC modeling and simulation on the aforementioned constructive principles delineated in the following Proposed Solution section.

1.3 RESEARCH OVERVIEW

Mathematical Optimization Framework The first real framework for ANN modeling, and still the preferred framework today, was first suggested by Werbos [54] but brought to fruition by, Rumelhart, Hinton, and Williams [47] and is concerned with determining the parameter values of a *single* network for a given data modeling problem of interest as an exercise in point optimization. This approach is fundamentally *white box* oriented, and the major technique of this approach is a calculus based analysis of a mathematical model of the network behavior in order to effect an error reduction via use of first (and often second) derivatives associated with the networks embedded non-linearities to gradient descend a hyper-dimensional error surface by *backpropagating* changes to the network parameters in order to reduce the network error as compared to the desired network output. Thus, the training of the network is tightly coupled to the mathematical representation of the networks,

response.

A further consideration is the need for *regularization* or noise training inhibition as the un-regularized technique risks tuning to the *particular* noise of the data sample being used for training. This approach also requires avoidance of certain deleterious behaviors associated with the explicit dependency on the derivatives of the non-linearities in the training algorithm. As was earlier stated, this is still the preferred framework by the ANN research community and is still the most active area of research with many variations and enhancements (mostly ad-hoc from our perspective) but suffers from a catalog of idiosyncratic deficiencies [30] calling its logical coherence into question.

Bayesian Backpropagation Framework A few researchers have departed from the main track of optimization-backpropagation and have begun to investigate more Bayesian probability based methods. In the early 1990s another major framework along this track was pioneered by Buntine and Weigend [7] known as *Bayesian Backpropagation*. This framework was still optimization based but drew upon some Bayesian principles of probability. Buntine and Weigend took a Maximum Likelihood based approach using a naive hyper-prior distribution for network parameters (no explicitly modeled covariances among network parameters). Further, their method used Laplace approximations around promising regions of the ostensible posterior to achieve a Gaussian mixture distribution, with the noise parameter integrated via an improper scale-invariant Jeffreys prior.

MacKay's Evidence Framework In the later 1990s, David J. C. MacKay developed [32, 33, 34] the *Evidence Framework*. In this framework, MacKay pursued the Empirical Bayes [2] methodology meaning that the Bayesian prior distribution was estimated from the data to be modeled using *hierarchical Bayes* techniques and equated to Maximum Likelihood estimation of the prior. This method made further assumptions about the analytical forms of both the posterior and prior in order to simplify some further approximations of each. It is by far the most commonly used and enhanced "Bayesian" framework reported on in the literature to this day; however, Bayesian purists consider Empirical Bayes to constitute a breach of Bayesian principles.

Mackay's approach is the contemporary "Bayesian" method of choice, finding its way, for example, in the Matlab ANN Toolbox under the moniker MacKay's Bayesian Regression (MBR). Quite recently, van Hinsbergen, van Lint and van Zuylen [57] used MacKay's Evidence framework to build an ANNC in the context of travel time predictions for highway traffic.

Neal and MacKay's Automatic Relevance Determination Framework Neal and MacKay's Automatic Relevance Determination Framework [35] took MacKay's Evidence Framework a step further by providing an individual hyperparameter for each network input so as to provide a basis for discriminating against the relevance of each network input individually, ostensibly to facilitate pruning the network of irrelevant inputs.

Neal's Bayesian Learning for Neural Network Framework Neal's Bayesian Learning for Neural Networks framework[41] relaxed analytical modeling of the posterior from MacKay's framework towards a more fully simulation based approach but still relied upon Empirical Bayes methods to handle the prior and thereby still retained a Maximum Likelihood optimization based core component. Neal structured his framework for infinite sized networks that converge asymptotically to Gaussian processes, retained ARD pruning, and used a complex MCMC sampler from the Quantum Chromo Dynamics (QCD) community.

Tippings's Framework Tipping [51] took ARD to its logical conclusion by providing a separate hyper parameter for *each actual weight*, thereby greatly increasing the complexity of the prior and the maximum likelihood pre-processing/tuning of the prior in Empirical Bayes fashion to the specific problem set data and attempted to prune the network at the individual network parameter level versus individual input level.

Lee Lee [31] attempted to constrain the search for useful models with a uniform prior via linear independence conditions on the design matrix in the linear system processing for the final (output) layer of the network. It is not clear whether this technique is limited to the use of identity functions in the final layer, but this is a potentially useful technique in our framework as well, although it is of secondary importance as a practical efficiency gain for sampling.

1.4 PROPOSED SOLUTION

Our approach to Artificial Neural Network (ANN) Committee (ANNC) modeling and simulation for both regression and classification is based on a more formal accounting for the logically underdetermined nature of candidate models that characterizes the modeling of data as a properly weighted superposition of decoupled deterministic and stochastic models. We model the residual uncertainty over a universe of candidate models without making reference to optimization. In so doing, we explore a methodology predicated on probability as uncertainty versus frequency. For this reason, an Objective Bayesian Probabilistic analysis is pursued as it refrains from any pre-tuning or pre-analysis of the Bayesian prior distribution with the data to be modeled.

The uncertainty over a universe of candidate models is due primarily to the presence of data that can only be characterized probabilistically or stochastically. Two approaches accounting for the uncertainty over a universe of candidate models proceed along the following conceptual lines: the first performs formal Bayesian predictive distribution modeling, and the second seeks an explicit separation between deterministic and stochastic models through a process of subtraction of determinism from the data to leave a stochastic residual. Simple statistical point estimates of the residue then provide for a relatively inexpensive prediction interval estimate for unencountered data based on the assumption of *homoscedasticity.*

1.5 RESEARCH OBJECTIVES

The principle research objectives of this study are to examine the feasibility of constructing robust probability distributions to predict the range and weighting of predictive outputs for ANNs for input data not used in modeling the parameters of the ANN. Several ancillary research goals are also indicated to support the primary goals:

- The feasibility of modeling the stochastic parameters of data using formal Bayesian methods.
- The feasibility of model based generalization via systematic extraction of the deterministic model component from data to comprise an explicit separation of deterministic and stochastic models which are the best inference as to the composition of the data.
- The feasibility of prior distribution regularization of ANN predictive performance.
- The feasibility of prior distribution generalization of ANN predictive performance.
- Do predictions based on formal predictive Bayesian probabilistic modeling out perform that of the more explicit attempt to separate stochastic and deterministic models to form the more inexpensive predictive intervals?

1.6 ANTICIPATED CONTRIBUTIONS

- A good performing methodology for formal Bayesian predictive distributions for ANNCs constructed from training data.

- A robust methodology for separation of deterministic and stochastic models from their superposition with good inexpensive predictive performance characteristics.

1.7 DOCUMENT OUTLINE

We summarize and critique the major features of these previously developed frameworks (both optimization and "Bayesian" backpropagation), then investigate ANN committee (ANNC) modeling and simulation instead on a logic based conceptual framework founded on plausible reasoning, information theory and a consistent probability calculus as a form of extended logic for *underdetermined*[2] analysis. Such a framework is fundamentally ANN *committee* oriented.

[2]In mathematical contexts where analytic expression are involved, such as the ubiquitous linear system $Ax = b$, whenever the matrix A is singular or the vector x has no unique solution, we are said to be dealing with an *ill posed* problem. Our proposed method does not expose us to this need for analytical inversion directly, and so we use the term *underdetermined*

CHAPTER 2

THEORETICAL BACKGROUND

2.1 NEURODES AND PERCEPTRONS

ANNs are networks of individual neurodes (artificial neurons). The first neurodes were originally inspired by biological research of the neurons in the human brain.

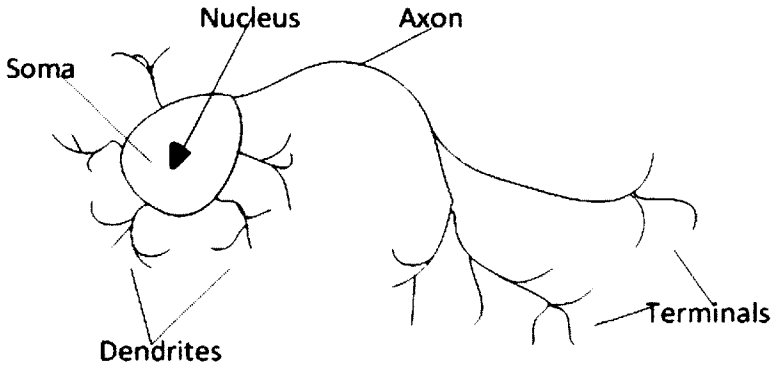


Fig. 1. Biological Neuron.

Threshold Logic Unit McCulloch and Pitts[40] developed a crude model of the biological neuron called the Threshold Logic Unit (TLU) in 1943 with two-state output and a step activation function based on a *threshold*, and is depicted in Figure (2).

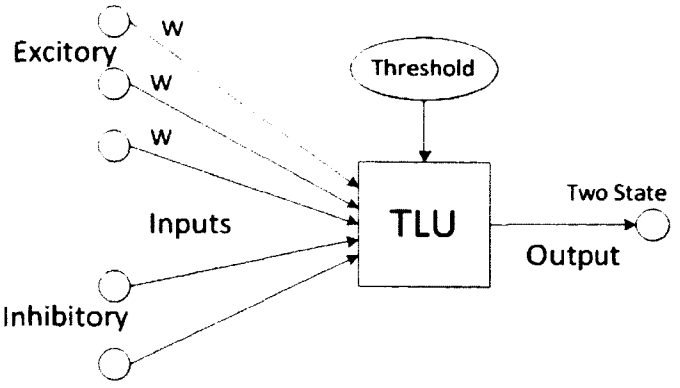


Fig. 2. McCullough-Pitts TLU.

It had the following operational characteristics:

- Binary outputs.
- Each neurode has a single fixed threshold.
- The neurode receives *weighted* or scaled inputs from excitatory synapses, all having identical positive weights ω.
- Inhibitory inputs have an absolute veto power over any excitatory inputs.
- At each time step the neurodes are updated by summing the weighted excitatory inputs and setting the output to 1 if and only if the sum is greater than or equal to the threshold and if the neurode receives no inhibitory input.

Hebbian Learning Electronics based simulation of neurodes was greatly stimulated by *Hebbian Learning* which was the concept that connection weights would change to model learning (pattern recall) and was introduced in 1949 by Hebb[22]. The first actual electronic simulation models of neurodes were studied in 1954 by

Farley and Clark [15, 16] (IBM) as well as Rochester, Holland, Haibit and Duda [43], in 1956.

The Perceptron In 1958 the first *Perceptron* that was a single neurode with multiple weighted inputs and step activation was developed by Rosenblatt [44, 45] who used Hebbian learning with thresholding and variable weights.

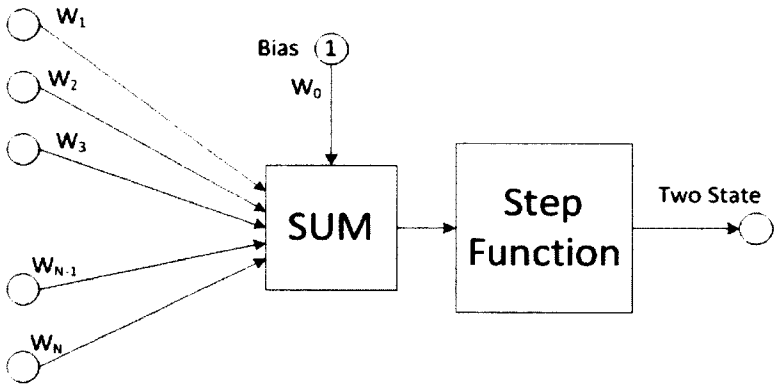


Fig. 3. Rosenblatt Perceptron.

It had the following operational characteristics:

- The weights and thresholds were not all identical.
- Weights can be positive or negative.
- There is no absolute inhibitory synapse.
- The neurodes have two-state output.
- Hebbian Learning.

The ADALINE In 1960 the ADaptive LINear Element or ADALINE by Widrow and Hoff [55, 36] (Stanford) was the first neurode to use non thresholded summing junctions for weighted non-binary inputs and introduced the *delta learning rule* for goal directed and *supervised learning*. The ADALINE modeled its output y as

$$y = \sum_{k=1} x_k \omega_k + b \tag{1}$$

for some input vector $\mathbf{x}$ and scaling or *weight vector* $\boldsymbol{\omega}$ plus a bias constant b. Defining $x_0 = 1$ and $\omega_0 = b$, we can write the ADALINE's output as simply $y = \mathbf{x} \cdot \boldsymbol{\omega}$.

Delta Learning Rule The delta learning rule is affected by changing the ADALINE's weight vector $\boldsymbol{\omega}$ according to the strategy

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \eta (\mathbf{y} - \mathbf{t}) \tag{2}$$

where $\mathbf{t}$ is the target or *training* output and therefore the supervision, and η is a *learning rate*. Thus is affected a form of gradient descent linear regression which converges to the least squares error. As is apparent from (1) the ADALINE suffered from the notable limitation of only being able to specify linear *classification boundaries*.

Minsky and Pappert's Criticism In 1969 Minsky and Pappert [38] were critical of the work to date exposing the linear-only classification boundary limitation of single layer perceptrons as too severe for important applications and research funding was derailed for some time after this.

2.2 MULTILAYER NETWORKS

In 1974 Werbos [54] published work that would be the genesis of the modern era of ANNs based on the *optimization backpropagation* algorithm for feed forward networks that showed how to extend the Widrow-Hoff delta rule to multiple layers, where some layers are embedded or *hidden* from the inputs. These hidden neurodes replaced the step function weighted and summed input transformations with continuous *non-linear* transforms. Subsequently in 1986, three independent research groups led by LeCun [29], Parker [42], and Rumelhart [47] developed *multi-layer perceptrons* with hidden layers and, using the Werbos inspired backpropagation algorithm, overcame the limitation of linear-only classification boundaries exposed by Minsky and Pappert, such that sufficiently complex networks are capable of determining arbitrarily complex classification boundaries (see Figure 4)

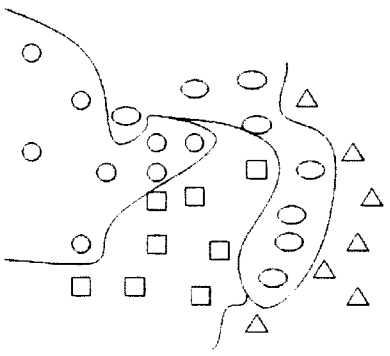


Fig. 4. Example Complex Non-Linear Classification Boundaries.

2.3 THREE LAYER PERCEPTRONS

Theorems from Cybenko[11], and Hornik, Stinchcombe and White [23] developed a few years later established that Three Layer Perceptron (TLP) networks with a weighted input layer, a hidden layer with continuous non-linearities, and a wighted output layer (with or without its own transform) of sufficient complexity were capable of approximating arbitrarily complex continuous non-linear functions to arbitrary accuracy [11, 23] and so began the modern era of ANNs of sufficient representational prowess for any/all non-linear regression and classification models. A typical example of this kind of network interconnection structure is depicted in Figure (5). Interest in this modeling technique has grown steadily to the present day.

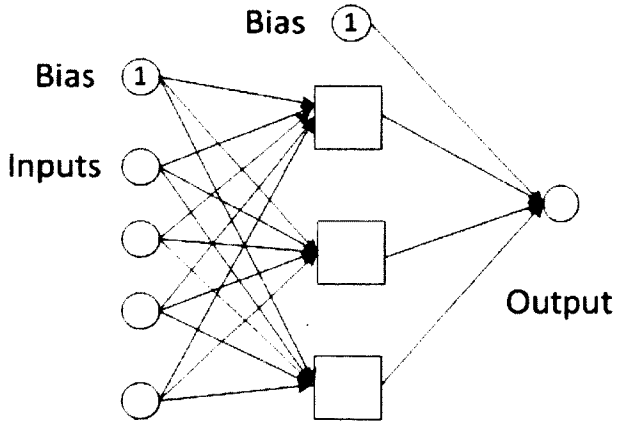


Fig. 5. Example Three Layer Feed Forward Perceptron Network.

Referring to Figure (5), the mapping of the N_i inputs through N_h hidden

neurodes to the output of the TLP can be written mathematically as

$$y_p = \nu_0 + \sum_{h=1}^{N_h} \nu_h \psi \left(\omega_{0h} + \sum_{i=1}^{N_i} \omega_{ih} x_{pi} \right) \tag{3}$$

where x_{pi} are the network input matrix components (given) for the i^{th} component of the p^{th} input *training pattern* and y_p is the network output for the p^{th} pattern, the W, V matrices with elements ω_{ih} and ν_h respectively are the network *weight* parameters for the input to hidden layer and the hidden to output layer respectively, and ψ are non-linear functions in the hidden layer. Weights ν_0 and ω_{0h} are for the bias inputs for the output and hidden layers respectively. In the case of a multidimensional output vector $\mathbf{y}$ versus a single output y, the network hidden to output scaling weight vector $\boldsymbol{\nu}$ becomes the matrix V with elements ν_{hk} such that

$$y_k = \nu_{0k} + \sum_{h=1}^{N_h} \nu_{hk} \psi \left(\omega_{0h} + \sum_{i=1}^{N_i} \omega_{ih} x_{pi} \right); \; k = 1, \cdots, N_o \tag{4}$$

where N_o is the number of elements in the output vector $\mathbf{y}$.

2.4 BAYESIAN PROBABILITY

Cox's theorem establishes the product rule of probability $P(H_1 H_2|B) = P(H_1|H_2 B)P(H_2|B)$ and the sum rule of probability $P(H|B) + P(\bar{H}|B) = 1$ as the unique rules of consistent probabilistic inference. The symmetry of the product

rule leads immediately to Bayes' Theorem:

$$P(H|D,B) = \frac{P_L(D|H)P_0(H|B)}{P(D)}. \tag{5}$$

In words the posterior probability of the hypothesis H conditioned on the data D is determined as the product of the independent probabilities (likelihood) of the data conditioned on the hypothesis and the prior probability of the hypothesis. All of this is considered to be conditioned on an overarching background B, especially the prior distribution. The factor $P(D)$ is the marginalized probability (using the sum rule) of the data over the universe of hypotheses and serves to normalize the distribution. Bayesian probability analysis is applied along two basic tracks.

In the first track known as parameter estimation, a single parameterized model is considered, and all possible values of its parameters are considered to be the universe of distinct actual models. In this case, the $P(D)$ factor is considered the probability for this model as a class. In the second track, Bayes' Theorem is used to grade model classes against one another. The upshot is that Bayes' Theorem is able in principle to probabilistically compare different model classes as well as different parameter vectors within each model class, thereby providing a complete probabilistic analysis for a given application.

Types of Bayesianism The Bayesian community has taken different approaches to dealing with the issue of posing the prior information required by the formal structure of Bayes' Rule. We understand the Bayesian prior as the *momentum term* in the formal structure of Bayesian inference. Here we survey these approaches with

some commentary:

- **Analytical**: Prior that is analytically *conjugate* to analytical posterior. An analytical convenience which is somewhat passe in the modern computational era.

- **Subjective**: Perfect encoding of background knowledge according to the analysts state of information about the application at hand.

- **Objective**: Uninformative or weakly informative priors. Designed to downplay influence of prior and is against the spirit of Bayes Theorem.

- **Empirical**: Prior estimated from data[2]. The opposite of Objective Bayes and that gives undue weight to prior artificially tuning it to the data, violating Bayes Theorem and risks artificially narrowing posterior and predictive distributions.

- **Jeffreys**: Ignorance/invariance priors. Ostensibly good where we are truly ignorant concerning background information. Often leads to improper (non-integrable) priors which may be a warning that this is a limiting condition and that there is no such thing as total ignorance.

- **Zellner**: Maximize influence of data. Another approach that is designed to downplay influence of prior and is against the spirit of Bayes Theorem.

- **Bernardo**: Reference priors[4] which maximize expected posterior information. An approach towards Bayesian "standardization." May be useful in our proposed framework due to the nature of ANNC modeling.

- **Jaynes**: *Least Bias* heuristic: Conditions priors by maximizing constrained information theoretic entropy. An overt approach to removing bias from prior using information theoretic ideas.

CHAPTER 3

RESEARCH REVIEW

3.1 MATHEMATICAL BACKPROPAGATION OPTIMIZATION

The first real framework for ANN modeling and still the preferred framework today was first suggested by Werbos [54] but later developed by, Rumelhart, Hinton, and Williams [47] and is concerned with determining the parameter values of a *single* network for a given data modeling problem of interest as an exercise in point optimization. The major feature of this approach is a calculus based analysis of the mathematical model (3).

The optimization based training regimen is to affect an error reduction of the *regularized* sum-of-squares (SSE) network error in the form of

$$RSSE \propto \sum_{p=1}^{N_p} (t_p - y_p)^2 + \lambda_\omega \sum_{\omega} \omega^2 + \lambda_\nu \sum_{\nu} \nu^2 \tag{6}$$

where the t_p is the (given) *training* pattern output and provides the supervised or *target*[1] for the network to model, and the individual weight sum terms are so called *regularizers* that penalize larger parameter values[2].

Training proceeds by making use of the first (and often second) derivatives associated with the non-linearities to *backpropagate* changes in the network weight

[1]except that they are contaminated with noise and the noise component of the training pattern output is not part of our desired target.

[2]Known as *parameter shrinkage* in the Statistics literature or *weight decay* in the ANN literature. They generally serve to *indirectly* inhibit training to the noise component of the target (training) outputs and also to provide for better *generalization* or performance on data not used in training.

parameters via a Widrow-Hoff Delta rule (2) of the form

$$\Delta(W,V)^{(0)} = -\alpha \nabla_{\omega,\nu} RSSE + \beta \Delta(W,V)^{(-1)} \tag{7}$$

such that the current change (designated by superscript (0)) is composed of the scaled current gradient of the RSSE where the gradient is performed in the hyperspace of all the elements in the V,W matrices, α is a *learning rate* parameter which scales the rapidity of the descent, and β is a momentum factor for the previous change (designated by superscript (-1)) designed to make the operation less likely to be trapped in a local minima.

This approach is still the preferred framework to date as evidenced by the vast amount of historical and current research literature devoted to it with many variations and enhancements (mostly ad-hoc from our perspective), but it suffers from a catalog of idiosyncratic deficiencies [30]. It should be further noted that this framework is not particularly committee oriented.

Inspection of equations (3)-(7) should convince the reader that this framework becomes rapidly more complex with size and/or as embedded layers are added[3]*, is increasingly susceptible to numerical instabilities, explicitly couples parameters with other parameters and their derivatives, and in general risks not scaling well for size or network complexity in light of its resultant explicit mathematical form.*[4]

[3]Considered highly desirable currently to support so called *deep learning.*

[4]See practically any text on Artificial Neural Networks for a full elaboration.

3.2 BAYESIAN BACKPROPAGATION

Buntine and Weigend [7] (B&W) took a different Maximum Likelihood based optimization approach with their *Bayesian Backpropagation* framework. This framework drew upon some Bayesian principles of probability to pose a Bayesian prior distribution for the network weight parameters ω (and similarly for ν) in hyper-parameterized form such that,

$$p(\omega) = \int_{\alpha} p(\omega|\alpha)\, p(\alpha)\, d\alpha = \int_{0}^{\infty} N\left(\omega|0, \alpha^{-1}\right) \left(\frac{1}{\alpha}\right) d\alpha \tag{8}$$

here using a zero-mean Gaussian with precision α and an improper[5] Jeffreys scale invariant hyper prior for the precision which is integrated out or *marginalized.* This results in the same simple naive[6] hyper-prior distribution for all network parameters. B&W used a noise-scaled SSE likelihood function with the noise parameter integrated or marginalized out to perform Maximum Likelihood analysis that, when combined with their prior distribution, facilitates a search for relative maxima in the resulting simplified posterior. They then used the Laplace approximation for the log of the posterior around those promising regions to form a Gaussian mixture approximation of $p(w, v|D)$ where D is the training data.

In our approach, we eliminate all dependence on hyper-priors, early marginalization of hyper-prior and noise parameters, and use of any/all convenient analytical approximations.

[5]We consider use of improper priors, though somewhat customary, to be a breach of principle based on the Cox axioms. We intend to use only proper prior distributions in our proposed framework. See [52], [12], [28] for discussion of this issue.

[6]No explicitly modeled covariances among parameters.

3.3 EVIDENCE FRAMEWORK

Continuing along the basic direction established by B&W, David J. C. MacKay developed [32, 33, 34] his *Evidence Framework*. In this framework, MacKay pursued the Empirical Bayes [2] methodology estimating the prior distribution from the data to be modeled using *hierarchical Bayes* techniques. MacKay used a single parameter hyper-prior in the form of

$$p(\omega|\alpha) \propto exp\left(-\frac{1}{2}\alpha \sum_{\omega} \omega^2\right) \tag{9}$$

(and similarly for v) and a likelihood function in the form of

$$p(D|\omega, \nu, \beta) \propto exp\left(-\frac{1}{2}\beta \cdot SSE(D, \omega, \nu)\right). \tag{10}$$

The network weights were first integrated out to set the hyper-parameters α, β s.t.,

$$p(D|\alpha', \beta') = \max\left\{\int p(D|w, v, \beta)\, p(w, v|\alpha)\, dw\ dv\right\} \tag{11}$$

and equated to a Maximum Likelihood estimation of both the prior in the form $p(w, v|\alpha', D)$ and the likelihood function as $p(D|w, v, \beta')$.

The critical thing to notice is that now the prior is dependent on the actual training data - including the particular noise sample in the data - in a way that directly conflicts with Bayesian probability and in particular Cox's Axioms. Edwards et al.[14] show that MacKay's method is prone to over-fitting.

MacKay's method is by far the most commonly used and enhanced "Bayesian" framework reported on in the literature and continues to be the method of choice[19][37][14][53][8] for Bayesian methods for ANNs. Recently, for example, van Hinsbergen, van Lint and van Zuylen [57] use MacKays Evidence framework to build an ANNC in the context of travel time predictions for highway traffic.

3.4 AUTOMATIC RELEVANCE DETERMINATION FRAMEWORK

Neal and MacKay's Automatic Relevance Determination Framework [35] (ARD) took MacKay's Evidence Framework a step further into more complex hyper-priors for the input to hidden unit network weights w by providing an individual hyper-parameter for each network input in the form

$$p(\omega|\alpha) \propto exp\left(-\frac{1}{2}\sum_{i,h}\alpha_i\omega_{i,h}^2\right) \tag{12}$$

so as to provide a basis for discriminating against the relevance of each network input individually, ostensibly to facilitate pruning the network from irrelevant inputs. A report by Husimer, Penny, and Roberts documents experiments where ARD fails[24].

3.5 EMPIRICAL BAYES FOR INFINITE NEURAL NETWORKS

Neal's framework for Bayesian Learning for Neural Networks [41] was oriented towards *infinite sized networks*. relaxed analytical modeling of the posterior from MacKay's framework - a sure improvement - but still relied upon an Empirical

Bayes core to handle the prior. However rather than doing maximum likelihood estimation of the prior as MacKay did, Neal tuned his prior to the data by including the priors' hyper-parameters in the MCMC sampling of the network's parameters in the network output function. He also retained ARD pruning and used a sophisticated MCMC sampler from the physics community.

While Neal relaxed analytical modeling of posterior, he used analytical conveniences (with hyper-parameters) for the prior (by his own admission) arguing for justification that the connection between prior and problem is sufficiently obscure. Neal's framework, while more sophisticated than MacKay's, has not seen significant further exploration by the ANN community perhaps because it is deemed too esoteric and lacks the practical and more manageable aspects of MacKay's framework.

3.6 EXTENDED ARD FRAMEWORK

Tipping[51] took the previous ARD frameworks to their logical conclusion by providing a hyper parameter for each actual weight in the form

$$p(\omega|\alpha) \propto exp\left(-\frac{1}{2}\sum_{k}\alpha_k\omega_k^2\right) \tag{13}$$

(and similarly for v) thereby greatly increasing the complexity of the prior and the maximum likelihood pre-processing/tuning of the prior to problem set data using Empirical Bayesian methods, and attempted to prune the network at the individual parameter level versus input level as had both MacKay and Neal.

3.7 OTHER RESEARCH

Lee [31] attempted to constrain the search for useful models with a uniform prior via linear independence conditions on the design matrix in the processing for the final (output) layer of the network. It is not clear whether this technique is limited to the use of identity functions in the final layer, but this is a potentially useful technique although it is oriented towards practical efficiency gains for sampling.

3.8 ISSUES WITH CURRENT APPROACHES

We list issues to be considered going forward for both backpropagation and Bayesian training techniques.

Mathematical Optimization Framework

- Must normalize data at all layers to avoid squashing function saturation and thus vanishing of derivatives
- Training of weights is coupled to other weights and their derivatives at gradient level (versus the joint probability level)
- Choice of non-linearity is dictated by idiosyncrasies of backpropagation mathematical details.
- No guarantee of convergence to desirable solution.
- Must deal with bias and variance of solutions.
- Solution is dependent on order of training patterns.

- Loses some data for training to support test set for early stopping.
- Best solutions require linear decorrelation of input vector components (e.g. PCA)
- Training efficacy is dependent on choice of squashing function
- Training is highly sensitive to mathematical details of network structure, which becomes progressively more complex and non-linear as network layers are added or expanded.
- Training instabilities can result in classification problems if represented at squashing function asymptotes.
- Solution is point solution oriented versus interval oriented.

Bayesian Frameworks

- Are Maximum Likelihood oriented
- Use Empirical Bayes pre-tuning of the prior distribution.
- Use analytic approximations to the posterior
- Analytically integrate hyperprior distributions to form prior distribution.

3.9 ISSUES NOT ADEQUATELY ADDRESSED

Polya's Plausible Reasoning George Polya was interested in modeling plausible reasoning to extend classical logical syllogisms, e.g. Modus Ponens $P \rightarrow Q; P \therefore Q$ and Modus Tollens $P \rightarrow Q; \neg Q \therefore \neg P$, for purposes of inductive reasoning. A good example is to consider a widespread conviction that $P \rightarrow Q; Q \therefore P$ which is in fact the fallacy of *affirming the consequent.* This brings up the issue of how to have a *calculus of evidence* for a model. Like Polya, we are instead interested in ideas such as $P \rightarrow Q; Q \therefore P$ becomes more/less plausible when compared to some $P1 \rightarrow Q$. More specifically, $P \rightarrow Q; Q \therefore P$ becomes more/less plausible compared to P1 as a function of the relative strength of $\rightarrow$, logical implication. How then also to characterize the strength of $\rightarrow$? I.e., implication ($\rightarrow$) is itself no longer a Boolean value. We are interested in a consistent calculus concerning evidence (in the form of data) for a model, where the remaining issues would seem to be how to make this rigorous and well founded logically.

Probability vs. Frequency While there a number of axiomatic or theoretical systems for probability[7], the actual history of probability has primarily reflected two differing perspectives known generally as frequentist and Bayesian[8]. It was actually the logical concept of probability that was earliest in the works of Bayes, Bernoulli, Laplace, Gauss, Karl Pearson, DeMorgan, and Borel[4]. P.S. Laplace and other physicists of his time achieved notable success with astronomical predictions using this logical probabilistic view. These works, however, were not axiomatized and

[7]e.g., Kolmogorov, de Finetti, Keynes to mention but a few

[8]We prefer logical and frequentist as the logical view actually preceded the frequentist view.

were considered too subjective to be a proper basis for probability[9] by mathematicians who subsequently developed the frequentist school that arose from a desire to achieve more mathematical rigor in the works of Cournot, Ellis, Boole, Venn, Fischer, Neyman, (Egon) Pearson, and Yates and Cochran (ANOVA).

The frequentist approach shifted the focus from state of investigator information from which to draw logical inferences, ostensibly to measurement of real frequency properties of a system under study and thereby essentially changed the definition of probability to one of physical frequency related to chance or randomness, and this view held sway from the late 1800s to the mid 1900s. In the early 20th century the original inferential viewpoint was defended primarily by Sir Harold Jeffreys.

By the time of the 1990s, powerful Monte Carlo sampling computational tools proved to give Bayesian methods in particular great assistance though classical statistics are still textbook orthodoxy and are what is most widely practiced.

Cox's Axioms of Consistency With a paper published in 1946 [9] and a follow up book in 1961 [10], R.T. Cox argued for the axiomatic basis for conducting inference that the degrees of plausible reasoning, when represented by real numbers, must conform to or be necessarily inconsistent. Cox was interested in knowing whether a calculus of consistent plausible reasoning in the real number system could be established, without assuming so, from elementary desiderata[10] of consistency which

[9]Henceforth we shall always mean the logical/inductive view with this term. When this term would imply *frequency*, we shall use that term.

[10]Latin: things wanted or needed

would also comport with Boolean algebra and common sense[11]. He was able to discover two resulting functional equations that when solved recovered the product and sum rules of conditional probability to within an arbitrary scale factor from which Bayes' Theorem follows directly.

Cox understood his result as identifying the uniquely consistent rules of probabilistic inference and considered his theorem as an extension to the ordinary propositional logic of Boole[12] covering all values in the closed interval [0,1], thereby establishing a comprehensive logic of propositional uncertainty which is to be applied for logically underdetermined problems. *Cox's results argue for the proper form of Bayesian probability which would preclude, for example, Empirical Bayes.* Jaynes understood Cox's result as the "most important conceptual contribution to the understanding of probability since the time of LaPlace".[26]

[11]Cox[10] argued that common sense is not to be dismissed but is used even in formal mathematical proofs and is therefore residual, and we are on safe grounds using it.

[12]Bayes Theorem reduces to classical logic (e.g. *modus ponens, modus tollens*) when the likelihood function degenerates to a delta function. In this case, we have $P(D|H') = \delta_{H'H}$; thus, $P(H'|D) \propto P(H')\delta_{H'H}$ or $D \rightarrow H; D \therefore H$ which is *modus ponens.*

CHAPTER 4

MARKOV CHAIN MONTE CARLO BAYESIAN PREDICTIVE FRAMEWORK FOR ARTIFICIAL NEURAL NETWORK COMMITTEE MODELING AND SIMULATION

The key principle upon which to establish this predictive framework is identification of properly weighting the outputs of different ANNs forming an ANNC in order to compute point and interval estimates of interest. As a logical matter, any modeling of output data from input data must necessarily be modeled deterministically, probabilistically, or a combination of both. Additionally, certain *flexible* residual researcher free choices must also be identified, and their impact on predictive modeling must be assessed.

4.1 NON LINEAR REGRESSION MODELING

We elucidate our concept for modeling the superposition of parameterized deterministic and stochastic models for non-linear regression modeling.

4.1.1 DETERMINISTIC ARTIFICIAL NEURAL NETWORK MODEL

Referring to Figure (5), for one-dimensional output (a simple scalar value) we model the anticipated deterministic portion of t, the training or supervisory value for an individual training pattern, using a Three Layer Perceptron (TLP) Artificial

Neural Network (ANN) model which conditionally maps an input vector $\mathbf{x}$ to an output y based on the values of the network parameter matrices W and V.

$$y = \nu_0 + \sum_{h=1}^{N_h} \nu_h \psi \left(\omega_{0h} + \sum_{i=1}^{N_i} \omega_{ih} x_{pi} \right). \tag{14}$$

We designate $y(\mathbf{x} \mid W, V)$ as the conditional output of the TLP when the input is $\mathbf{x}$ and values of the network weight parameters are W and V. We limit our analysis to a single scaler output for the network since other output elements in a higher dimensioned output vector would be computed in parallel with their own distinct set of parameters (c.f., equation (4) for an arbitrary TLP) and, if correlated with the chosen single output element, might serve better as additional informative inputs.

Many choices for hidden neurode continuous non-linearities are available. Generally, a *sigmoidal* shape is recommended, and we choose the *inverse tangent* for all work going forward (see Figure (6)). Inverse tangent is desirable primarily because it ranges equally on either side of zero.

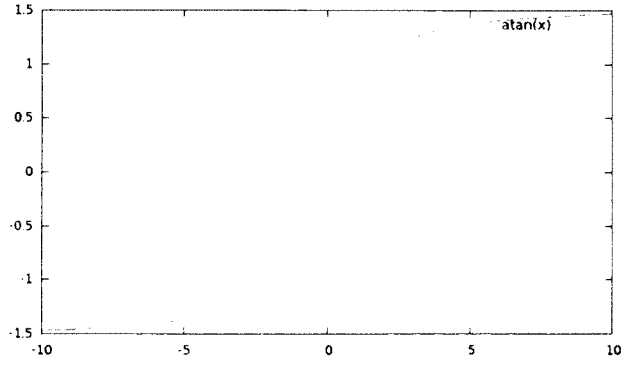


Fig. 6. Inverse Tangent.

4.1.2 STOCHASTIC MODEL

By adding a parametric stochastic model $\epsilon(\sigma_\epsilon)$ to the deterministic TLP model (14), we now seek to model t as

$$t \sim y(\mathbf{x} \mid W, V) + \epsilon(\sigma_\epsilon). \tag{15}$$

Unless we have specific information concerning the nature of the anticipated stochastic portion of the data, according to the Principle of Maximum Entropy, we should always choose the highest entropy distribution possible to model this portion of the data; accordingly, we choose an iid, uncorrelated, zero-mean Gaussian white noise model with a parametric variance, such that

$$\epsilon(\sigma_\epsilon) \backsim \phi\left(0, \sigma_{bl}^2\right) \tag{16}$$

where ϕ is the Gaussian pdf, and our task now is to determine good values for W,V,σ_{bl}.

4.1.3 MAXIMUM ENTROPY BASED LIKELIHOOD FUNCTION

For a collection of N_t training patterns from (15) and (16) we can now write t, y as vectors over that collection and now have

$$\mathbf{t} \sim \mathbf{y} + \phi(\mathbf{0}, \mathbf{\Sigma}) \tag{17}$$

where $\mathbf{\Sigma}$ is now the full covariance matrix over the dimension of the vector $\mathbf{t}$. With the *individually and identically distributed* assumption the covariance matrix has no off-diagonal components, the diagonal elements are the same value, and this becomes

$$\mathbf{t} - \mathbf{y} \sim \prod_{n=1}^{N_t} \phi(0, \sigma^2). \tag{18}$$

Equation (18) is known as a *naive* model since it does not account for possible parameter covariances and is another way of specifying a WGN model. The reason for choosing WGN is twofold: (1) the ANN is expected to model all deterministic mappings from the input data to the output data including any serial correlations and (2) the Gaussian distribution has the highest entropy of any two parameter distributions and is motivated by application of the PME as that noise figure which *assumes the least information* about the stochastic residue.

For one-dimensional ANN output, equation (18) is to be used in Bayes Rule (5) as the likelihood function

$$P_L(\mathbf{t}|\mathbf{y}, \sigma^2) = \phi(\mathbf{t} - \mathbf{y} \mid \mathbf{0}, \sigma^2) \tag{19}$$

along with some prior distribution, the prior being determined as discussed subsequently.

Let us define $\theta \equiv (W, V)$, i.e. the parameters for just the deterministic portion of the total model such that the total model parameters are now written as

$\Theta = (\theta, \sigma_{bl})$. The likelihood (19) can be written as

$$P(\mathbf{t} \mid \Theta, \mathbf{x}) = \prod_{k=1}^{N_t} \phi \left[t_k - y_k (x_k \mid \theta) \mid 0, \sigma_{bl}^2 \right] \tag{20}$$

where N_t is the number of training input patterns.

4.1.4 BAYESIAN ANN PRIOR DISTRIBUTIONS

We argued previously that the Cox Axioms of Consistency dictate a form of Bayesian Analysis that precludes the Empirical Bayes methodology used in previous frameworks in which Bayesian techniques were utilized since that method conditions the Bayesian prior on the data to be modeled. We argued the proper form of Bayesianism is that of equation(5), where the relationship between prior and likelihood is structured as one of independence. This effectively shifts the thrust of the ensuing analysis from optimization to logical inference.

ANN Weight Parameter Subjective Prior

We argue that the obscure *explanatory* relationship between ANN weight parameter with the input to output functional mapping effectively precludes any serious notion of a Bayesian subjective prior based ostensibly on background knowledge of the chosen data and network details.

ANN Weight Parameter Reference Prior

We utilize a *Bayesian ANN reference prior* based on our interpretation of the meaning of the Gaussian Distribution as characterizing essential background information concerning the characteristic or expected scale within which the proper values of the network parameters are to be found. Figure (7) depicts a Jeffreys scale invariant prior and a Gaussian prior; both mean = 0. The Gaussian curve has precision β, and we note that this Gaussian curve is above the Jeffreys curve in a characteristic scale region symmetric on both sides of the mean of zero. The Jeffreys prior is strictly improper (due to an infinite singularity at the origin) and is generally considered to represent the limit of total scale ignorance.

We therefore interpret the Gaussian distribution as giving extra prior weighting to a certain characteristic scale within which we expect to find the network's parameter values. Our experience has shown that certain characteristic scales are apropos for scale network parameters which tend to become smaller as the network complexity grows, whereas location parameters should be modeled with relatively broad uninformative priors. We therefore construct prior distributions based on these principles.

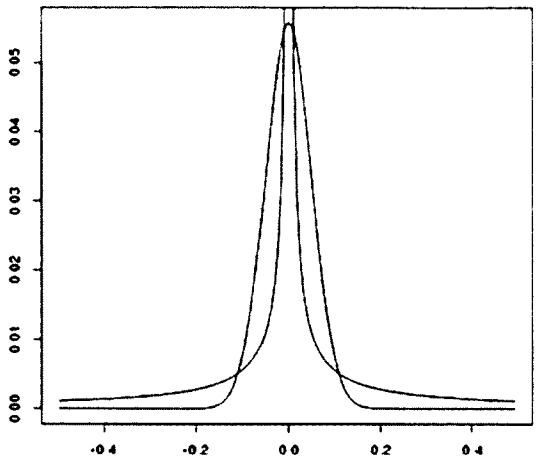


Fig. 7. Jeffreys vs. Gaussian.

Stochastic Model Prior

Prior distributions for the model parameter σ_{bl} of the stochastic model portion of the composite model discussed in section 4.1.2, require a different approach than the ANN weight parameters. These are necessarily scale parameters and are positive only. For this reason, we model the prior distribution for σ_{bl} with Gamma distributions of either a broad flavor to express high uncertainty concerning the stochastic residual or a narrow flavor if we wish to make the model consider a more definite residual stochastic model and express at least some notion of a proper weighting of a noise parameter prior that has not degenerated to a Jeffreys distribution. Figure 8 shows that a chosen Gamma distribution will express a priori skepticism (less than Jeffreys) both concerning certain small and certain large values of σ_{bl} and a priori confidence in a certain region (greater than Jeffreys).

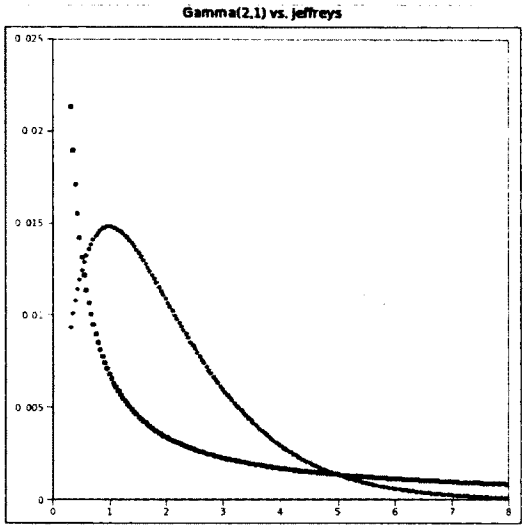


Fig. 8. Jeffreys vs. Gamma(2,1).

4.1.5 MODEL PARAMETER POSTERIOR DISTRIBUTION:

We proceed to infer all model parameters using Bayes' Rule which here becomes

$$P(\Theta|\ \mathbf{t},\mathbf{x}) \propto P(\mathbf{t}\mid \Theta,\mathbf{x})\, P_0(\Theta)\,. \tag{21}$$

These distributions are a means to an end, namely providing the proper averaging of committee models to compose the EDCM and also the Bayesian predictive distribution and credible intervals.

4.1.6 MCMC HYPOTHESIS SAMPLING

Our model parameter posterior probability expression (21) represents a collection or *committee* of models weighted probabilistically. In determining an appropriate ANNC to model given data, the crucial consideration is determination of an adequate number of candidate ANN models over which to evaluate the range of uncertainty.

In multidimensional and typically high-dimensional parameter spaces it is necessary to utilize advanced sampling techniques from Markov Chain Monte Carlo (MCMC) since the volume of the solution space increases exponentially with network size.

ANN Weight Parameter Sampling For ANN weight parameter sampling, we use a conventional Metropolis Markov Chain Monte Carlo sampling procedure whereby a proposed new parameter vector element value for say $\theta_k^{(1)}$ is produced from the current one $\theta_k^{(0)}$ by drawing a sample from a Gaussian distribution as a *proposal distribution* of a variance specified *for each individual parameter* and where the mean value for the proposal distribution is always the current value. The new values are accepted with probability:

$$\min\left[1, \frac{P\left(\theta_k^{(1)}\right)}{P\left(\theta_k^{(0)}\right)}\right] \tag{22}$$

where $P\left(\theta_k^{(1)}\right)$ and $P\left(\theta_k^{(1)}\right)$ are determined using Bayes Rule (21) for each θ_k individually. In cases where the proposed value is rejected, the current value is retained as the "new" value.

Stochastic Model Parameter Sampling For ANN stochastic model parameter σ_{bl} sampling, we also use a Metropolis Markov Chain Monte Carlo sampling procedure that is similar to *importance sampling.* In this case, we used a fixed Gamma distribution - the same as the prior distribution - to determine a proposed new value for σ_{bl} and then apply the Metropolis acceptance/rejection step (22). This procedure is conceptually similar to *importance sampling.*

4.1.7 EXPECTED COMMITTEE DETERMINISTIC MODEL

Formally our Bayesian expected value for an ANNC output $\mathbf{y}'$ for some input $\mathbf{x}'$ is determined as

$$E\left[\mathbf{y}' \mid \mathbf{x}', \mathbf{x}, \mathbf{t}\right] = \int_{\Theta} P\left(\Theta \mid \mathbf{t}, \mathbf{x}\right) \mathbf{y}\left(\mathbf{x}' \mid \Theta\right). \tag{23}$$

However using MCMC, we calculate this expectation by summing over the posterior Markov Chain (of non warm-up length N_{mc}) as

$$E\left[\mathbf{y}' \mid \mathbf{x}', \mathbf{x}, \mathbf{t}\right] \simeq \frac{1}{N_{mc}} \sum_{n=1}^{N_{mc}} \mathbf{y}_n\left(\mathbf{x}' \mid \Theta_n\right) \tag{24}$$

to form the Expected Committee Deterministic Model (ECDM).

4.1.8 PARAMETER MARGINAL DISTRIBUTIONS

Using MCMC, we can calculate a marginalized distribution for (say) the parameter σ_{bl} as an estimate of the true value of σ_ϵ characterizing the stochastic portion of the given data $\mathbf{t}$ by summing over the MCMC chain using the posterior values in the chain for (21) as

$$P\left(\sigma_{bl} \mid \mathbf{t}, \mathbf{x}\right) \simeq \sum_{\theta} P\left(\Theta \mid \mathbf{t}, \mathbf{x}\right) \tag{25}$$

followed by normalization to achieve formal interval information.. Distributions for the components of the parameter θ would be similar.

4.1.9 PARAMETER EXPECTED VALUES (STOCHASTIC MODEL)

Using MCMC, we can calculate an expectation for model parameters such as σ_{bl} over the chain as

$$E\left[\sigma_{bl}\right] \simeq \frac{1}{N_{mc}} \sum_{n=1}^{N_{mc}} \left(\sigma_{bl}\right)_n \tag{26}$$

and serve as point estimates. Expectations for the components of the parameter θ would be similar.

4.2 PREDICTIVE MODELING

The central thrust of our effort is in evaluating our thesis, namely that our method provides effective predictive modeling. Both Banks, et. al.[3], and Avrill and Law[1] appear to agree that "*The most definitive test of a simulation model's validity is to establish that its output data closely resemble the output data that would be expected from the actual (proposed) system*"[1]

4.2.1 NON-LINEAR REGRESSION

Non linear regression is principally about regression statistics and this is primarily concerned with statistical properties of the residual data after the regression fit has been subtracted. Good statistical characterization of this residual then provides for prediction of new output data from new inputs as they relate probabilistically to the EDCM output for the new input.

Bayesian Predictive Distributions (BPD)

We model formal Bayesian Predictive Distributions capable of predicting the probabilistic excursion from an EDCM for previously unseen data according to formal Bayesian principles and then make output predictions for some arbitrary input x' to our parameterized deterministic model (14). Due to the uncertainty in the network deterministic parameters θ and due to the presence of stochasticity in the given data $\mathbf{t}$, we make this prediction in the form of a probability distribution $P(y' \mid x', (\mathbf{t}, \mathbf{x}))$ as properly reflective of the uncertainty in Θ, where in essence we are making a predictive model for x' based on the given data $(\mathbf{t}, \mathbf{x})$ and our choices of network model structure, Bayesian likelihood model, and Bayesian prior distribution. This is most directly accomplished by formally marginalizing (integrating out) the parameter vector Θ from the joint probability distribution for Θ and arbitrary outputs y' from arbitrary inputs x' as

$$\begin{aligned} P(y' \mid x', (\mathbf{t}, \mathbf{x})) &= \int_{\Theta} dF_{\Theta} P(y', \Theta \mid x', \mathbf{x}, \mathbf{t}) \\ &= \int_{\Theta} d(\Theta) P(\Theta \mid \mathbf{x}, \mathbf{t}) P(y' \mid \Theta\ x', \mathbf{x}, \mathbf{t}) \\ &= N_{\Theta} \int_{\Theta} d(\Theta) P_0(\Theta) P(\mathbf{t} \mid \Theta, \mathbf{x}) P(y' \mid \Theta\ x', \mathbf{x}, \mathbf{t}) \\ &= N_{\Theta} \int_{\Theta} d(\Theta) P_0(\Theta) \prod_{k=1}^{N_t} \phi\left[t_k - y_k(x_k \mid \theta) \mid 0, \sigma_{bl}^2\right] \\ &\qquad \times \phi\left[y' - y(x' \mid \theta) \mid 0, \sigma_{bl}\right] \end{aligned} \tag{27}$$

where F_{Θ} is the distribution function for Θ and we have made use of Bayes Rule (21) (the learned posterior distribution), or the conditional likelihood (20) for the

given data and the prior distribution for the parameters (an investigator free choice), and N_Θ is a normalization constant. This is the integral of our posterior weighted likelihood function.

Using MCMC, we calculate approximate (though asymptotically exact) Bayesian predictive distributions for an arbitrary output y' from an arbitrary input x' based on (27) as

$$P\left(y'|x',(\mathbf{t},\mathbf{x})\right) \simeq \frac{1}{N_{mc}} \sum_{n=1}^{N_{mc}} \phi\left(\left[y' - y\left(x' \mid \theta_n\right)\right] \mid 0, \left(\sigma_{bl}\right)_n\right). \tag{28}$$

Using these distributions, Bayesian Credible Intervals (BCI) of 90%, 95%, 99% are then determined as intervals of the indicated percentage of the total probability computed by equation (28) .

Stochastic Residue Method (SRM)

Our SRM approach takes a different approach and seeks reliable predictions on unseen data by achieving a suitable stochastic residue as a form of inexpensive interval estimation based on point estimates of the descriptive statistics of the residual.

In this approach, we subtract the ECDM (24) from the given data model $\mathbf{t}$ in the training phase to form the *stochastic residue* model $\mathbf{t} - E\left[\mathbf{y} \mid \mathbf{x}\right]$. This results in a practical separation of the deterministic from the stochastic portion of the given data.

The descriptive statistics - if they are suitably suggestive - are then used to

frame a zero mean Gaussian stochastic model of the indicated standard deviation σ_{srm} which is *then applied to the residual of the data withheld from training* and reserved for predictive performance tests. If the associated skew γ_{srm} and kurtosis κ_{srm} are not too severe, there should be good predictive agreement between these data and the predictive residual.

It should be noted that the SRM approach is not coupled in any way to our method of producing an EDCM using Bayesian probability and MCMC. For this reason, it is applicable in principle for any trained network regardless of training method, Bayesian or optimization, and will be the primary basis for making comparisons to solutions provided by the Matlab ANN Toolkit.

4.2.2 TLP NON-LINEAR CLASSIFICATION

Classification problems are mappings to discrete labels that are typically encoded numerically, where the input data may either be continuous or discrete.

Binary Classification

For binary classification data, we have the training output data matrix T as a collection of binary encoded values either 0 or 1. The dimension of the output vector for each input pattern is now $N_o = 1$, so that the matrices T and Y are now just vectors. These binary values are interpreted as degenerate probabilities of observing the training output data. For this reason, there is no additive noise in the training data, and there is no apparent need for an additive stochastic model such as equation(16). We therefore model the given degenerate probability matrix T (41) as

our general deterministic ANN model Y (40) such that the probability of observing (say) the first element of T, which is t_1, is the corresponding element of Y namely y_1 where y_1 is interpreted using the Bernoulli Distribution such that

$$P(t_1 \mid y_1) = y_1^{t_1} (1 - y_1)^{1-t_1} \tag{29}$$

as now there is no parameterized likelihood function. Here the network output Y is interpreted as the network's answer as to the probability that the input X (35) corresponds to the output T given the Bayesian posterior probability distribution over the *deterministic only* ANN parameter set θ equation (21) in the modified form

$$P(\theta \mid T, X) \propto P(T \mid X) P_0(\theta). \tag{30}$$

For the predictive data, Y is by convention the networks predictive probability that the proper classification of the input is the class represented by the value of '1'. For the entire training output matrix T, the likelihood of observing T for input X is interpreted using the Binomial Distribution as:

$$P(T \mid \theta, X) = \prod_{k=1}^{N_t} y_k^{t_k} (1 - y_k)^{1-t_k} \tag{31}$$

which on a log scale is

$$log_b(P(T \mid \theta. X)) = \sum_{k=1}^{N_t} t_k log_b(y_k) + (1 - t_k) log_b(1 - y_k) \tag{32}$$

and is the so called *cross-entropy error function*[6] of the backpropagation learning method. **Here it is used in our methodology as a true Bayesian probabilistic likelihood and not an error function.** *It is worth repeating that unlike the backpropagation method, we are not doing optimization (minimization) but are using MCMC to build a computed joint Bayesian posterior probability distribution* of the parameters associated with the ANN deterministic output (θ) in order to construct the resulting ANNC EDCM and which represents direct modeling of probabilities.

One of Many Classification

For more than two classes, a so-called *1-of-C* classification problem, the likelihood function becomes

$$P(T \mid \theta, \mathbf{x}) = \prod_{c=1}^{C} \prod_{n=1}^{N_t} Y_{nc}^{T_{nc}}, \tag{33}$$

and the activations of the output layer, which are the right hand side of (40), are subsequently *softmaxed* [39] to form a *winner-take-all* non-linearity to indicate the winning class such that the appropriate element of the network output vector y assumes the value '1' while all others assume the value '0' to be used with the given training data T (41) to compute (33). This is accomplished by passing each row of the matrix Y (40) with elements y_{pc}, $c = 1, \cdots, C$ through the transformation

$$y_{pc} \leftarrow \frac{exp(E(y_{pc}))}{\sum_{c'=1}^{C} exp(E(y_{pc'}))} \tag{34}$$

which results in one of the output vector elements assuming the value one and all other elements the value zero as the network prediction for the matrix T (41). This is known as a *1-of-C binary coding scheme.* Differences between this prediction and the given training data (T), which is also *1-of-C binary* encoded, are then used to compute the *misclassification* where 5% is the customary figure of merit.

4.3 GENERAL DETERMINISTIC TLP SIMULATION

Our method is applicable for any feed forward ANN regardless of structure, in a supervised learning context for both regression and classification. Further, it benefits from the virtue that it is generally insensitive to the complexities of network structure unlike conventional backpropagation training where the analytical expressions for the derivatives of especially the more deeply embedded neurodes or the more numerous embedded neurodes becomes progressively more complex and entangled with other parameters and their derivatives (c.f. section 3.1) not to mention progressive susceptibility to the vagaries of numerical computation.

Since it has been established that all ANNs are equivalent in modeling capabilities to TLPs [6], we limit our study to TLPs for practical reasons. In the case of an ANN with TLP structure, we utilize a convenient matrix structured computation as follows. Let N_t be the number of training input vectors, N_i the length of the input vector, N_h the number of neurodes in the hidden layer, N_o the length of the output vector.

Quasi Matrix Formulation We organize the simulation of an arbitrarily complex TLP operating on arbitrarily complex input and output data as a quasi-matrix computation according to the following algorithm:

1. Organize the N_t given input data (row) vectors $\mathbf{x}$ each of length N_i for each pattern into a matrix of dimension $N_t \times N_i$. Next, prepend a column of 1s to this matrix to represent the bias input for each pattern into the following matrix with dimensions $N_t \times (N_i + 1)$ such that:

$$X \equiv \begin{pmatrix} 1 & x_{11} & \cdots & x_{1N_i} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_{N_t 1} & \cdots & x_{N_t N_i} \end{pmatrix}. \tag{35}$$

2. Organize the input (with biases) to neurode scaling weights ω_{ih} into the following matrix with dimensions $(N_i + 1) \times N_h$:

$$W \equiv \begin{pmatrix} \omega_{01} & \cdots & \omega_{0N_h} \\ \cdots & \cdots & \cdots \\ \omega_{N_i 1} & \cdots & \omega_{N_i N_h} \end{pmatrix}. \tag{36}$$

3. Form the hidden layer *activation* matrix with dimensions $N_t \times N_h$ as:

$$A = XW. \tag{37}$$

4. Process each element of the matrix A through the chosen neurode *activation*

function $\psi(\cdots)$ to form the matrix A_ψ.

5. Prepend a column of 1s (again to represent the bias) to the processed (*squashed*) matrix A_ψ to form the output layer activation matrix Q with dimensions $N_t \times (N_h + 1)$ such that:

$$Q(X|W) \equiv \begin{pmatrix} 1 & q_{11} & \cdots & q_{1N_h} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & q_{N_t 1} & \cdots & q_{N_t N_h} \end{pmatrix}. \tag{38}$$

6. Organize the neurode output (with biases) to output scaling weights ν_{ho} into the following matrix with dimensions $(N_h + 1) \times N_o$ such that:

$$V \equiv \begin{pmatrix} \nu_{01} & \cdots & \nu_{0N_o} \\ \cdots & \cdots & \cdots \\ \nu_{N_h 1} & \cdots & \nu_{N_h N_o} \end{pmatrix}. \tag{39}$$

7. Form the output matrix of the ANN with dimensions $N_t \times N_o$ as:

$$Y(X|W,V) = QV. \tag{40}$$

It is straightforward to verify that this prescription yields the TLP output (3) for all components of the output vectors for all input training patterns as the matrix Y.

8. Finally, organize the N_t given output data (row) vectors $\mathbf{t}$ each of length N_o

for each pattern into the matrix T of dimension $N_t \times N_o$ such that:

$$T \equiv \begin{pmatrix} t_{11} & \cdots & t_{1N_o} \\ \cdots & \cdots & \cdots \\ t_{N_p1} & \cdots & t_{N_pN_o} \end{pmatrix}. \tag{41}$$

4.4 CONSTRUCTION

The primary tools utilized for this effort were:

- the R computing environment for statistical and probabilistic modeling as well as algorithmic development,
- the Matlab ANN Toolbox for optimization backpropagation based ANN modeling,
- the Gnumeric spreadsheet for general use,
- the C computing language for our system's implementation on a Linux Operating System platform,
- the Numerical Recipes in C source code library for general numerical computational support,
- the Gnuplot environment for graphics and data visualization.

CHAPTER 5

EVALUATIVE METHODOLOGY AND EXPERIMENTAL DESIGN

General Remarks Simulation output analysis generally requires us to determine the essential nature of the simulation at hand, namely terminating versus non-terminating and then for non-terminating, one of steady state, cyclic, or other for output measures[1]. In the case of ANN/TLP supervised learning simulation, there is no natural termination criteria other than one derived from the learning procedure itself, namely an arbitrary run length or achievement of steady state values for an error figure in the case of backpropagation or sufficient exploration of a parameter hyperspace in the case of MCMC.

This places our effort in the non-terminating MCMC category where the size of the MC must be adequate for representing the sought after invariant joint probability distribution of the network parameters so that the committee model of the network deterministic output properly represents the effect of the supervised learning process, including residual uncertainty as to the proper deterministic model.

As no sure method exists to determine MCMC convergence[5], these considerations indicate the use of *face-validation* to examine the process diagrams for the MCMC parameter sampling to assure us that the parameter space has been adequately sampled via attainment of steady state variation of network parameter values as reflected by the MC. Other strategies for MCMC generally involve

use of sampling refinements to inhibit excessive *random walk* behavior, for example Metropolis-Hasting sampling versus Metropolis sampling.

5.1 OVERVIEW

Since our thesis leads to a new conceptual framework for ANN modeling and simulation, it seems appropriate to conduct a two phased evaluation. Firstly, we propose a *learning* empirical evaluation for how the system is performing concerning the goal of learning from data. This would seem to align well with *Measures of Performance* (MOP) which are oriented towards measuring what the system is actually doing in terms of learning performance.

Secondly, we propose a *predictive* empirical evaluation concerning how effectively the system uses what has been learned to make predictions. This would seem to align well with *Measures of Effectiveness* (MOE) which are oriented towards measuring how effective the modeled system is towards its ultimate goal of modeling given data and predictions of not previously encountered data.

We employ cross validation approaches for BPD, SRM, and Matlab SRM within the context of K-Factorial test matrices for each to test learning and prediction responses for a range of settings and to explore some basic sensitivity testing and to see if more optimal learning and prediction settings are indicated for both regression and classification.

Generalization To achieve effective predictive modeling with ANNs, it is important to achieve good *generalization.* Traditional ANN training methods have generally sought generalization by pursuing good *regularization*, which is an overt attempt to inhibit training to the stochastic portion of the data albeit *indirectly* by parameter shrinkage modifications to the objective function of the optimization based training approach. (c.f. Section 3.1). We prefer a more direct approach by attempting to model both deterministic and stochastic parameters of the combined model using MCMC model parameter sampling and model parameter set evaluations as Bayesian hypotheses; we assess the capabilities of our system to achieve good generalization by examining both the training and predictive performance of our system under varying conditions and the connection between the two.

For each factor combination design point in the K-factorial design matrix, a separate cross validation with a fixed number of repeated trials is performed as outlined below.

5.2 CROSS VALIDATION

We use *repeated random sub-sampling* cross validation for both our method and the Matlab ANN Toolbox on common data modeling problems and then compare results. This method repeatedly randomly splits the dataset into training and validation data (also test data for Matlab) for each trial. For each such split, the model is trained on the training data, and predictive accuracy is assessed on the validation data as discussed in section 5.3.

The statistics for each measured quantity in the cross validation are then calculated across the trials. We list below the cross validation trial steps common to both our method and ANN training/testing with the Matlab ANN Toolbox.

Common

- The given problem data is randomly shuffled for each trial so that training, testing (if required), and predictive sets are unique for each trial.
- Training data for each trial is always completely in sample, whereas that for validation (and testing if needed) is always out of sample.
- For each trial, an EDCM is produced that is a strictly deterministic model.
- For each trial, a residuals data set for both training and prediction are constructed by subtraction of the EDCM training solution from the (shuffled) training data for the trial. In this manner, trial stochastic models of length N_t for the training portion, and N_p for the predictive portion are produced.
- Each cross validation typically consists of 1000 independent trials but may be more or less depending on circumstances.
- Cross validation measurements are computed as averages across trials. When interval information for the measurement is reported, it is calculated as ($3\times$ Standard Error of the Mean) for the set of all trials of the cross validation experiment to form a 99% confidence interval for the measurement.

We list below the cross validation trial steps used for BPFANN for both BPD and SRM.

BPFANN BPD/SRM

- There is no requirement to form a test residue data set.
- Pre warming of MCMC chains is used so that no data is collected or analyzed on cold chains. Chain pre-warming is done by training on non-partitioned data with non aggressive stochastic model parameter settings so as to produce only a loose fit.
- Data for each trial is partitioned into training and predictive subsets according to a 85%/15% partitioning rule in order to provide a counterpart match to the Matlab Toolbox defaults. Note that here we absorb that portion of data normally used to support *early stopping* backpropagation optimization (15%) into the training (learning) portion as an advantage of our method since it does not require an independent test set to aid in the training phase.
- Each trial runs for 1000 MCMC transitions for each parameter on the pre-warmed MCMC chain.
- For each trial, the resultant EDCM is constructed from the Markov Chain produced in the trial.
- For each trial, a residuals data set is produced as the difference between the given data and the trial EDCM.

Matlab SRM

- Data for each trial is partitioned into training, testing and predictive subsets according to the 70%/15%/15% default partitioning rule needed to support backpropagation optimization training with *early stopping generalization.*

- For each trial the resultant EDCM is the natural point solution of the backpropagation method chosen. It is not a true committee model, but we retain that nomenclature for easy comparison with BPFANN.

- For each trial, a residual's data set is produced as the difference between the given data and the trial EDCM.

5.3 CROSS VALIDATION MEASUREMENTS

We delineate the measurements and ensuing analysis of the training and predictive outcomes for cross validation trials.

5.3.1 NON-LINEAR REGRESSION MODELING

ANN modeling is a form of non-linear regression modeling; therefore, our measurements are concerned with statistical characterization of the EDCM models and associated data residual, where this residual is defined as the subtraction of the EDCM model from the given data to form the stochastic residue as discussed in the mechanics of our method Chapter 4, and is true both for our Bayesian and also Matlab ANN Toolbox solution methodologies. Proper probabilistic characterization of

the training residual comprises our MOP. Similarly, proper probabilistic characterization of the predictive residual comprises our MOE. The measured data per cross validation trial that comprise our MOP are as follows.

EDCM Measurements

- Mean Square Error of the training residual considered as putative zero mean data,
- Pearson's R of the EDCM compared to the training portion of the given data,
- The first ten (or fewer) autocorrelation coefficients of training/predictive residuals,
- The first four statistical moments (mean, standard deviation, skew, kurtosis) of training/predictive residuals.

Auto Correlation Coefficients We also measure the auto-correlation coefficients of the residual data according to the formula

$$R(\tau) = \frac{\mathrm{E}[(X_t - \mu)(X_{t+\tau} - \mu)]}{\sigma^2} \tag{42}$$

for lag τ and interpret these data in conjunction with the statistical moments described as to the quality of the ANN data modeling effort for a given example.

Mean Square Error We also include a measurement of Mean Square Error (MSE) as a datum of historical interest to the ANN community and especially to facilitate direct comparison to the results achieved by the Matlab ANN Toolbox. MSE will

prove to be an interesting measurement since it's minimization in training is not the proper goal as explained in Section 3.1. We include it to provide a common measurement with the optimization backpropagation approach as it is implemented in the Matlab ANN Toolbox. From equation (18) this is calculated as

$$\mathrm{MSE} = \frac{1}{N_t}\sum_{n=1}^{N_t}(t_n - y_n)^2. \tag{43}$$

Pearson's R Pearson's R is included as a common measurement with the Matlab ANN Toolbox, and from equation (18) is calculated according to the prescription

$$r = \frac{\sum_{n=1}^{N_t}(t_n - \mathrm{E}[t])(y_n - \mathrm{E}[y])}{\sqrt{\sum_{n=1}^{N_t}(t_n - \mathrm{E}[t])^2}\sqrt{\sum_{n=1}^{N_t}(y_n - \mathrm{E}[y])^2}}. \tag{44}$$

Residual Statistical Moments We measure the first four statistical moments of the data residual obtained after subtraction of the EDCM from the training data, namely mean (μ_{srm}), standard deviation (σ_{srm}), skew (γ_{srm}), and kurtosis (κ_{srm}). For SRM, the primary purpose is to test conformance of the predictive residual to a zero-mean WGN signature as characterized by the value of σ_{srm} determined by the training portion of the data only as described earlier in section 4.2.1. From equation (18) our formulae for these data are

$$\mu_{srm} = \frac{1}{N_t}\sum_{n=1}^{Nt}(t_n - y_n) = \mathrm{E}[t - y] \tag{45}$$

$$\sigma_{srm} = \sqrt{\frac{1}{N_t - 1}\sum_{n=1}^{N_t}((t_n - y_n) - \mu_{srm})^2} \tag{46}$$

$$\gamma_{srm} = \frac{\frac{1}{N_t}\sum_{n=1}^{N_t}((t_n - y_n) - \mu_{srm})^3}{\left(\frac{1}{N_t}\sum_{n=1}^{N_t}((t_n - y_n) - \mu_{srm})^2\right)^{3/2}} \tag{47}$$

$$\kappa_{srm} = \frac{\mathrm{E}[((t - y) - \mu_{srm})^4]}{(\mathrm{E}[((t - y) - \mu_{srm})^2])^2} - 3. \tag{48}$$

SRM Measurements

- Percentages of residual data that occur *within* the $\pm 1\sigma_{srm}$, $\pm 2\sigma_{srm}$, $\pm 3\sigma_{srm}$, $\pm 4\sigma_{srm}$, $\pm 5\sigma_{srm}$, $\pm 6\sigma_{srm}$ excursion envelopes from zero where the value of σ_{srm} are those produced by the training procedure. These values are to be compared against the known figures for a Gaussian Distribution of the same variance. *This is the primary evaluation criteria for the Stochastic Residue Method.*

The *residual mean* μ_{srm} is expected to be near zero, and we interpret the standard deviation σ_{srm} as the SRM estimation of the parameter σ_ϵ in equation (16). Any measured *excess* skew γ_{srm} and kurtosis κ_{srm} are interpreted as defects in the deterministic modeling intended to achieve a zero-mean WGN residue. Since finite samples from zero-mean WGN data generators typically indicate some measured skew and kurtosis, we particularly look for *excess* skew and kurtosis.

Bayesian Credible Intervals

Formal Bayesian Probability Distribution evaluation proceeds along the lines described in the Bayesian Predictive Distributions Section (4.2.1), where for each trial for *learning* evaluation we restrict our attention to the data used for training

and, likewise, for predictive evaluation we use the data withheld from training.

Practical matters dictate that discretized linear grids be centered on the EDCM value abscissa locations and extend on either side along the abscissa axis at each EDCM ordinate point sufficiently to test probability values for the anticipated excursions from the EDCM. Probability values along this test grid are then assessed using equation (28), and simple count statistics are accumulated as to the number of values in the training or predictive data portion that are included in the indicated 90%, 95%, 99% credible intervals.

As a practical matter, these key values of the credible intervals are typically located between predictive grid points and are linearly interpolated. Statistics are accumulated on the size of these intervals over the complete set of cross validation trials. For BPD the predictive residual is used to test conformance with the Bayesian Credible Intervals also determined by the training portion of the data only as discussed in Section 4.2.1. Measured data includes

- Percentages of residual data that occur *within* the 90%, 95%, and 99% Bayesian *credible intervals* using equation (28) for both training and predictive partitions of the given data set for BPD predictions.

- Widths of the 90%, 95%, and 99% Bayesian credible intervals measured in the same units as the network output.

Matlab ANN Toolbox

In this case, we proceed with the same measurements described for EDCM and SRM except that the Matlab EDCM is computed as a simple arithmetic average

committee model over all point solutions produced per individual trial.

5.3.2 CLASSIFICATION

Classification measurements are conceptually simple as we make a direct head-to-head comparison of our network's output data to the given data to directly compute the misclassification percentages for both the training and predictive data portions as indicated in Section 4.2.2.

5.4 K-FACTORIAL DESIGN

We utilize a K-Factorial trial stimulation and analysis as elaborated in Law and Kelton[1], such that a separate cross validation is performed, as discussed previously, for each design point in the K-Factorial design matrix. The various purposes of this approach are:

- network committee performance assessment diagnostic,
- sensitivity testing,
- relate MOE and MOP.

The general recommendation for factor settings is to space them such as not to depart from a linear system response between factor setting changes.

5.4.1 2^3 FACTORIAL DESIGN FACTORS FOR BPFANN NON-LINEAR REGRESSION

The three factors for the design are as follows:

1. N_h as the number of hidden neurodes in a TLP to represent size or complexity of the network (Section 4.3). This factor is listed simply as the number of neurodes in the hidden layer.

2. Prior distributions for the ANN weight matrices W,V in the form of the standard deviations of zero mean Gaussians (Section 4.1.4). This factor is listed as a triple. The triple consists of the zero-mean Gaussian distributions with specified standard deviations for the network parameters in the following order: ω_{0h} where h is arbitrary, ω_{ih} where $i > 0$ and h is arbitrary, and finally ν_h where h is arbitrary. For example, the triple $(\phi(0,1), \phi(0,10^{-1}), \phi(0,10))$ specifies zero-mean Gaussians of standard deviations of 1,10^{-1}, and 100 respectively. The actual prior then is the product of these so specified distributions as appropriate over the actual number of parameters in the matrices W,V. See section 4.1.4.

3. Gamma distribution shape parameters for the stochastic model parameter σ_{bl} of the Likelihood function (Sections 4.1.3, 4.1.4).

2^3 Factorial Design for EDCM Measurements

Figure 9 is the *Design Matrix* for the EDCM measurements for our 2^3 factorial design where the '-' setting for each factor is intended to be the smaller numerical value for each factor and the '+' for the higher where in general these values are problem dependent. For the responses R_k, we have listed the measurements elaborated on in Section 5.3 for MSE,R, μ_{srm}, σ_{srm}, γ_{srm}, κ_{srm}, and ACC.

	A	B	C	D	E	F	G	H	I
1	Nh	W,V	Gamma	MSE	R	Mean	StDv	Skew	Kurt
2	-	-	-	R1	R1	R1	R1	R1	R1
3	+	-	-	R2	R2	R2	R2	R2	R2
4	-	+	-	R3	R3	R3	R3	R3	R3
5	+	+	-	R4	R4	R4	R4	R4	R4
6	-	-	+	R5	R5	R5	R5	R5	R5
7	+	-	+	R6	R6	R6	R6	R6	R6
8	-	+	+	R7	R7	R7	R7	R7	R7
9	+	+	+	R8	R8	R8	R8	R8	R8

Fig. 9. 2^3 EDCM Measures K-Factorial Matrix.

For Figure 10, the number of columns are for the ACC responses for the indicated lag value.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Nh	W,V	Gamma	1	2	3	4	5	6	7	8	9	10
2	-	-	-	R1	R1	R1	R1	R1	R1	R1	R1	R1	R1
3	+	-	-	R2	R2	R2	R2	R2	R2	R2	R2	R2	R2
4	-	+	-	R3	R3	R3	R3	R3	R3	R3	R3	R3	R3
5	+	+	-	R4	R4	R4	R4	R4	R4	R4	R4	R4	R4
6	-	-	+	R5	R5	R5	R5	R5	R5	R5	R5	R5	R5
7	+	-	+	R6	R6	R6	R6	R6	R6	R6	R6	R6	R6
8	-	+	+	R7	R7	R7	R7	R7	R7	R7	R7	R7	R7
9	+	+	+	R8	R8	R8	R8	R8	R8	R8	R8	R8	R8

Fig. 10. 2^3 EDCM ACC Measures K-Factorial Matrix.

2^3 Factorial Design for SRM Measurements

Figure 11 is the design matrix for the SRM measurements for our 2^3 SRM factorial design for the responses R_k, we have listed the measurements elaborated on in Section 5.3 for SRM. Note that we have also included measurements for the 3×Standard-Error-of-the-Mean (designated as 3SEM) resulting from the trial statistics for the cross validation to characterize a 99% confidence interval for the associated

measurements.

	A	B	C	D	E	F	G	H	I	J	K
1	Nh	W,V	Gamma	1s	3SEM	2s	3SEM	3s	4s	5s	6s
2	-	-	-	R1		R1		R1	R1	R1	R1
3	+	-	-	R2		R2		R2	R2	R2	R2
4	-	+	-	R3		R3		R3	R3	R3	R3
5	+	+	-	R4		R4		R4	R4	R4	R4
6	-	-	+	R5		R5		R5	R5	R5	R5
7	+	-	+	R6		R6		R6	R6	R6	R6
8	-	+	+	R7		R7		R7	R7	R7	R7
9	+	+	+	R8		R8		R8	R8	R8	R8

Fig. 11. 2^3 SRM K-Factorial Matrix.

In Figure 11, the column designated as '1s' is for the data inclusion percentage response for $\pm 1 \times \sigma_{srm}$ about the EDCM and likewise for columns '2s', ..., '6s'.

2^3 Factorial Design for BPD Measurements

Figure 12 is the design matrix for the BPD measurements for our 2^3 BPD factorial design where for the responses R_k we have listed the measurements elaborated on in Section 5.3 for BPD. Note that we have also included measurements for the 3×Standard-Error-of-the-Mean (designated as 3SEM) resulting from the trial statistics for the cross validation to characterize a 99% confidence interval for the associated measurements. Columns designated by percentages are for residual data inclusion, and those listed as 'CI' are for the widths of the Bayesian credible intervals.

	A	B	C	D	E	F	G	H	I	J	K	L
1										CI	CI	CI
2	Nh	W,V	Gamma	90%	3SEM	95%	3SEM	99%	3MSE	90%	95%	99%
3	-	-	-	R1		R1		R1		R1	R1	R1
4	+	-	-	R2		R2		R2		R2	R2	R2
5	-	+	-	R3		R3		R3		R3	R3	R3
6	+	+	-	R4		R4		R4		R4	R4	R4
7	-	-	+	R5		R5		R5		R5	R5	R5
8	+	-	+	R6		R6		R6		R6	R6	R6
9	-	+	+	R7		R7		R7		R7	R7	R7
10	+	+	+	R8		R8		R8		R8	R8	R8

Fig. 12. 2^3 BPD K-Factorial Matrix.

2^3 Factorial Effects

Single factor effects for each of the three factors upon each of the responses are calculated according to the conventional prescription:

$$e_1 = \frac{1}{4}(R_2 - R_1 + R_4 - R_3 + R_6 - R_5 + R_8 - R_7) \tag{49}$$

$$e_2 = \frac{1}{4}(R_3 - R_1 + R_4 - R_2 + R_7 - R_5 + R_8 - R_6) \tag{50}$$

$$e_3 = \frac{1}{4}(R_5 - R_1 + R_6 - R_2 + R_7 - R_3 + R_8 - R_4)\,. \tag{51}$$

Two factor effects for each combination of two of the three factors upon each of the responses are calculated according to the conventional prescription:

$$e_{12} = \frac{1}{2}\left(\frac{1}{2}(R_4 - R_3 + R_8 - R_7) - \frac{1}{2}(R_2 - R_1 + R_6 - R_5)\right) \tag{52}$$

$$e_{13} = \frac{1}{2}\left(\frac{1}{2}(R_6 - R_5 + R_8 - R_7) - \frac{1}{2}(R_2 - R_1 + R_4 - R_3)\right) \tag{53}$$

$$e_{23} = \frac{1}{2}\left(\frac{1}{2}(R_7 - R_5 + R_8 - R_6) - \frac{1}{2}(R_3 - R_1 + R_4 - R_2)\right). \tag{54}$$

Finally, the only three factor combination interaction is

$$e_{123} = \frac{1}{2}\left[\frac{1}{2}(R_8 - R_7) - \frac{1}{2}(R_6 - R_5) - \frac{1}{2}(R_4 - R_3) + \frac{1}{2}(R_2 - R_1)\right]. \quad (55)$$

Interpretations of these effects is straightforward, namely:

- e_k: the average change in response due to factor k moving from '-' setting to '+'.
- e_{kj}: difference between the average responses when k and j have the same sign and when they have opposite signs. ($e_{kj} = e_{jk}$)
- e_{kjm}: three factor interaction as half the difference between the average two factor interaction between factors k, j when m is '+' and when m is '-'. (e_{kjm} has the same value for all permutations of the labels k, j, m). *The literature notes that the interpretation of this effect can be obscure.*

5.4.2 2^2 FACTORIAL DESIGN FOR MATLAB ANN TOOLBOX NONLINEAR REGRESSION

The results from ANN training/testing from the Matlab ANN Toolbox will serve as a comparative baseline of best-of-breed commercial software. The default methodology of the Toolbox is to train the network via optimization backpropagation by dividing the data into a 70%/15%/15% split for training/testing/prediction. 'Testing' in this context generally means monitoring the network error to stop training when it begins to increase after reaching a minimum, a technique known as *early*

stopping that is endemic to optimization based methods such as the backpropagation training method. Also included is a training method derived from MacKay's Evidence (MBR) framework and should provide at least some opportunity for comparisons to that method.

Matlab ANN Toolbox performance will also be compared to our system's MOE/MOP using the EDCM and SRM measurements (section 5.3), except that the design matrix for Matlab is a 2^2 Factorial Design where the design factors are:

- Network Complexity as the number of hidden neurodes (N_h^-, N_h^+).
- Training Method as one of Scaled Conjugate Gradient or a derivative of Mackay's Bayesian Regression (MBR) Framework. SCG = '-', MBR = '+'.

Figures 13, 14, and 15 are the design matrices for the EDCM and SRM measurements for Matlab results.

	A	B	C	D	E	F	G	H
1	Nh	Mthd	MSE	R	Mean	StDv	Skew	Kurt
2	-	-	R1	R1	R1	R1	R1	R1
3	+	-	R2	R2	R2	R2	R2	R2
4	-	+	R3	R3	R3	R3	R3	R3
5	+	+	R4	R4	R4	R4	R4	R4

Fig. 13. 2^2 EDCM K-Factorial Matrix.

	A	B	C	D	E	F	G	H	I	J
1	Nh	Mthd	1s	3SEM	2s	3SEM	3s	4s	5s	6s
2	-	-	R1		R1		R1	R1	R1	R1
3	+	-	R2		R2		R2	R2	R2	R2
4	-	+	R3		R3		R3	R3	R3	R3
5	+	+	R4		R4		R4	R4	R4	R4

Fig. 14. 2^2 SRM K-Factorial Matrix.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Nh	Mthd	1	2	3	4	5	6	7	8	9	10
2	-	-	R1	R1	R1	R1	R1	R1	R1	R1	R1	R1
3	+	-	R2	R2	R2	R2	R2	R2	R2	R2	R2	R2
4	-	+	R3	R3	R3	R3	R3	R3	R3	R3	R3	R3
5	+	+	R4	R4	R4	R4	R4	R4	R4	R4	R4	R4

Fig. 15. 2^2 ACC K-Factorial Matrix.

Single factor effects for each of the two factors upon each of the responses are calculated according to the conventional prescription:

$$e_1 = \frac{1}{2}\left(R_2 - R_1 + R_4 - R_3\right) \tag{56}$$

$$e_2 = \frac{1}{2}\left(R_3 - R_1 + R_4 - R_2\right). \tag{57}$$

Two factor effects for the combination of the two factors upon each of the responses are calculated according to the conventional prescription:

$$e_{12} = \frac{1}{2}\left(\frac{1}{2}\left(R_4 - R_3\right) - \frac{1}{2}\left(R_2 - R_1\right)\right). \tag{58}$$

Interpretations of these effects is straightforward, namely:

- e_k: the average change in response due to factor k moving from '-' setting to '+'.

- e_{kj}: difference between the average responses when k and j have the same sign and when they have opposite signs. ($e_{kj} = e_{jk}$)

5.4.3 NON-LINEAR REGRESSION DESIGN POINT RATIONALE

BPDFANN: The rationale for the design point choices for BPDFANN are as follows:

- Choices for N_h represent less complex versus more complex networks for a modeling application.

- The '+' prior setting is for a relatively broader and less discriminating prior for $\boldsymbol{\theta}$ and may be chosen to represent a uniform prior, while the '-' prior setting is to be more discriminating (localized) and may represent our notion of a general reference prior based on broad experience with our system.

- Priors for σ_{bl} are intended to coerce relatively less/more aggressive noise training inhibition, and express at least some notion of a proper weighting for a noise parameter prior that has not degenerated completely to a Jeffreys scale ignorant distribution.

Matlab: The rationale for the design point choices for Matlab are as follows:

- Choices for N_h represent less complex versus more complex networks for a modeling application.

- The '-' method setting is recommended for general use by Matlab for traditional optimization style training. The '+' setting is for MacKay's Bayesian Regularization (section 3.3) and should provide an interesting comparison to our system.

5.4.4 2^2 FACTORIAL DESIGN FOR BPFANN CLASSIFICATION

Figure 16 is the *Design Matrix* for the misclassification percentage measurements for our 2^2 factorial design where the '-' setting for each factor is intended to be the smaller numerical value for each factor and the '+' for the higher where in general these values are problem dependent. For the responses R_k, we have listed the measurements elaborated on in Section 5.3.2. Factor effects will also be assessed. The rationale for the design choices is the same as discussed above.

	A	B	C
1	Nh	W,V	MisClass
2	-	-	R1
3	+	-	R2
4	-	+	R3
5	+	+	R4

Fig. 16. 2^2 BPFANN Classification K-Factorial Matrix.

5.4.5 2^2 FACTORIAL DESIGN FOR MATLAB ANN TOOLBOX CLASSIFICATION

Figure 17 is the *Design Matrix* for the misclassification percentage measurements for our 2^2 factorial design where the '-' setting for each factor is intended to be the smaller numerical value for each factor and the '+' for the higher where in general these values are problem dependent. For the responses R_k, we have listed the measurements elaborated on in Section 5.3.2. Factor effects will also be assessed. The rationale for the design choices is the same as discussed above.

	A	B	C
1	Nh	Mthd	MisClass
2	-	-	R1
3	+	-	R2
4	-	+	R3
5	+	+	R4

Fig. 17. 2^2 Matlab Classification K-Factorial Matrix.

5.5 DATA

Use of synthetic data sets (via simulation) is the ideal way to accomplish our aims since the correct answers are in hand (ground truth), and data not used in the construction phase of the ANNC modeling can be generated at will. In addition, we test against selected problems studied in the research literature to include those available from the Univ. Calif. at Irvine Machine Learning Database (UCIMLDB).

CHAPTER 6

EXPERIMENTAL CASE STUDY RESULTS AND ANALYSIS

6.1 NON-LINEAR REGRESSION CASE STUDY I

In this case study, we analyze a data set produced by simulation from a known deterministic function

$$f(x) = 10^{-5}x^3 \sin(0.2x) \exp\left(\frac{x}{150}\right) \tag{59}$$

and a known stochastic model $\epsilon \leftarrow \phi(\mu = 0, \sigma^2 = 2)$ that is random variates drawn from a Gaussian distribution with mean 0 and standard deviation of two. We generate this Noisy math Function (NMF) training output data $\mathbf{t}$ of length 200 such that

$$t_x = f(x) + \epsilon \tag{60}$$

where $x = -100, \cdots, 100$. With the 85%/15% train/predict partitioning this is 170/30 patterns respectively. The first four descriptive statistical moments (mean, standard deviation, skew, kurtosis) of the true stochastic portion are: $\mu_\epsilon = 0.0155$, $\sigma_\epsilon = 1.88$, $\gamma_\epsilon = 0.039$, $\kappa_\epsilon = 0.0126$. Ground truth MSE for those figures is 3.54. Both noised and de-noised curves are shown in Figure 18.

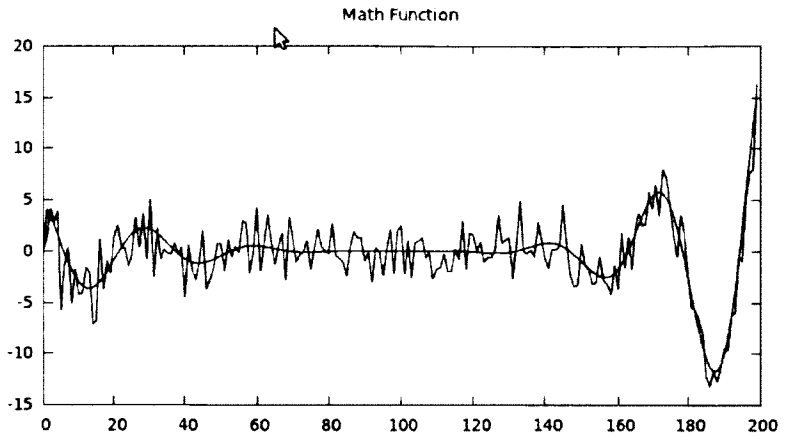


Fig. 18. Noisy Math Function.

Simulation generated data affords the opportunity to assess system performance in light of known quantities, also known as *ground truth.*

6.1.1 BAYESIAN CREDIBLE INTERVALS

From a Standard Normal Distribution for the given values of the true stochastic data (from a known Gaussian distribution), we may calculate the sizes of Bayesian Credible Intervals (BCIs) for 90%, 95%, 99% actual data inclusion about the EDCM that we might expect from good BCI modeling as follows:

- 90% BCI width $= 2 \times 1.65 \times \sigma_\epsilon = 6.2$,
- 95% BCI width $= 2 \times 2 \times \sigma_\epsilon = 7.52$,
- 99% BCI width $= 2 \times 2.575 \times \sigma_\epsilon = 9.68$.

6.1.2 CROSS VALIDATION

We perform a cross validation study on this data for 1000 trials for both BPFANN (BPD/SRM) and Matlab (SRM) with the parameter setting matrices for both the 2^3 and 2^2 cross validations as listed in Tables 1 and 2 as per sections 5.4.2 and 5.4.1.

Table. 1: 2^3 NMF Factorial BPFANN Design Point Settings

Design Point	N_h	$P_0(\boldsymbol{\theta})$	$P_0(\sigma_{bl}) = \Gamma(shape, scale)$
(- - -)	3	$\phi(0,1), \phi(0,10^{-1}), \phi(0,10)$	$\Gamma(2, 0.5)$
(+ - -)	5	$\phi(0,1), \phi(0,10^{-1}), \phi(0,10)$	$\Gamma(2, 0.5)$
(- + -)	3	$\phi(0,100), \phi(0,100), \phi(0,100)$	$\Gamma(2, 0.5)$
(+ + -)	5	$\phi(0,100), \phi(0,100), \phi(0,100)$	$\Gamma(2, 0.5)$
(- - +)	3	$\phi(0,1), \phi(0,10^{-1}), \phi(0,10)$	$\Gamma(2, 1.5)$
(+ - +)	5	$\phi(0,1), \phi(0,10^{-1}), \phi(0,10)$	$\Gamma(2, 1.5)$
(- + +)	3	$\phi(0,100), \phi(0,100), \phi(0,100)$	$\Gamma(2, 1.5)$
(+ + +)	5	$\phi(0,100), \phi(0,100), \phi(0,100)$	$\Gamma(2, 1.5)$

Table. 2: 2^2 NMF Factorial Matlab Design Point Settings

Design Point	N_h	Training Method
(- -)	3	SCG
(+ -)	5	SCG
(- +)	3	MBR
(+ +)	5	MBR

Cross Validation Responses

The SRM and BPD 2^3 cross validation results for BPFANN are in Figures 19, 20, 21, and 22, and the results for Matlab SRM are in Figures 28, and 29. For Figure 19, columns F and M were added as described previously, since ground truth data is available for this particular case study.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Train			3.54							Predict						
2	Nh	W,V	Gamma	MSE	R	GT R	Mean	StDv	Skew	Kurt	MSE	R	GT R	Mean	StDv	Skew	Kurt
3	-	-	-	3.61	0.84	0.8	0.02	1.90	0.1	−0.1	3.69	0.909	1	0.09	1.85	0.1	−0.2
4	+	-	-	**2.97**	0.87	**0.61**	−0.04	1.72	0.0	0.4	4.32	0.892	0.98	0.12	2.04	**0.6**	**1.2**
5	-	+	-	3.58	0.84	0.8	0.02	1.89	0.1	−0.1	3.87	0.908	1	0.13	1.89	0.0	−0.3
6	+	+	-	**2.99**	0.87	**0.62**	−0.03	1.73	0.0	0.4	4.18	0.9	0.98	0.08	2.00	**0.7**	**1.4**
7	-	-	+	3.61	0.84	0.8	0.05	1.89	0.1	−0.3	4.37	0.897	0.99	0.20	2.06	0.3	−0.2
8	+	-	+	3.24	0.86	0.74	0.01	1.80	0.1	−0.2	3.66	0.912	0.99	0.21	1.87	0.3	−0.3
9	-	+	+	3.59	0.84	0.8	0.04	1.89	0.1	−0.3	4.5	0.891	0.99	0.24	2.07	0.3	0.0
10	+	+	+	3.22	0.86	0.74	0.03	1.79	0.1	−0.2	3.68	0.912	0.74	0.13	1.89	0.2	−0.2
11			**e1**	**−0.5**	**0.02**	**−0.12**	−0.04	**−0.13**	−0.07	0.3	−0.1	0.00	**−0.07**	0.0	0.0	**0.3**	**0.7**
12			**e2**	0.0	0.00	0.00	0.01	0.00	0.01	−0.03	0.0	0.00	**−0.06**	0.0	0.0	0.0	0.1
13			**e3**	0.1	−0.01	0.07	0.04	0.03	0.04	−0.42	0.0	0.00	**−0.06**	0.1	0.0	−0.1	−0.7
14			**e12**	0.0	0.00	0.00	0.01	0.00	0.00	−0.01	−0.1	0.00	**−0.06**	0.0	0.0	0.0	0.0
15			**e13**	0.1	−0.01	0.06	0.02	0.04	0.10	−0.17	**−0.6**	**0.02**	**−0.06**	0.0	−0.2	**−0.4**	**−0.9**
16			**e23**	0.0	0.00	0.00	0.00	0.00	0.01	0.00	0.0	0.00	**−0.06**	0.0	0.0	0.0	0.0
17			**e123**	0.0	0.00	0.00	0.00	−0.01	0.01	0.01	0.1	0.00	**−0.06**	0.0	0.0	0.0	−0.1

Fig. 19. Noisy Math Function EDCM 2^3 Cross Validation Results.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Train			ACC									
2	Nh	W,V	Gamma	1	2	3	4	5	6	7	8	9	10
3	-	-	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	+	-	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	-	+	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	+	+	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	-	-	+	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	+	-	+	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	-	+	+	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	+	+	+	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11			e1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12			e2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13			e3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14			e12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15			e13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16			e23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17			e123	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18													
19	Predict			ACC									
20	Nh	W,V	Gamma	1	2	3	4	5	6	7	8	9	10
21	-	-	-	0.0	−0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	+	-	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23	-	+	-	0.0	0.0	0.0	0.0	0.0	−0.1	0.0	0.0	0.0	−0.1
24	+	+	-	0.1	0.0	−0.1	0.0	0.0	0.0	0.0	0.0	−0.1	0.0
25	-	-	+	0.0	0.0	0.0	0.0	0.0	0.0	−0.1	0.0	−0.1	0.0
26	+	-	+	0.0	0.0	0.0	0.0	−0.1	0.0	0.0	0.1	0.1	0.0
27	-	+	+	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	+	+	+	0.0	0.0	0.0	−0.1	0.0	0.0	0.0	0.1	0.0	0.0
29			e1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30			e2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31			e3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
32			e12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
33			e13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
34			e23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35			e123	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig. 20. Noisy Math Function ACC 2^3 Cross Validation Results.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1				Train								Predict							
2				68%		96%		99.7%	100%	100%	100%	68%		96%		99.7%	100%	100%	100%
3	Nh	W,V	Gamma	1s	3SEM	2s	3SEM	3s	4s	5s	6s	1s	3SEM	2s	3SEM	3s	4s	5s	6s
4	-	-	-	68%	0.3%	95%	0.2%	100%	100%	100%	100%	67%	2.6%	95%	1.3%	100%	100%	100%	100%
5	+	-	-	71%	0.4%	95%	0.2%	100%	100%	100%	100%	67%	2.6%	**90%**	1.3%	**96%**	100%	100%	100%
6	-	+	-	68%	0.4%	**94%**	0.2%	100%	100%	100%	100%	66%	2.6%	94%	1.4%	99%	100%	100%	100%
7	+	+	-	71%	0.4%	95%	0.2%	100%	100%	100%	100%	68%	2.6%	**91%**	1.3%	**96%**	100%	100%	100%
8	-	-	+	**65%**	0.4%	97%	0.2%	100%	100%	100%	100%	**62%**	2.6%	**93%**	1.2%	100%	100%	100%	100%
9	+	-	+	70%	0.4%	95%	0.2%	100%	100%	100%	100%	67%	2.4%	**93%**	1.2%	100%	100%	100%	100%
10	-	+	+	**65%**	0.4%	97%	0.2%	100%	100%	100%	100%	**63%**	2.9%	**93%**	1.1%	99%	100%	100%	100%
11	+	+	+	70%	0.4%	96%	0.2%	100%	100%	100%	100%	67%	2.4%	**92%**	1.2%	100%	100%	100%	100%
12			e1	**4%**		-1%		-0%	0%	0%	0%	**3%**		**-2%**		-1%	0%	0%	0%
13			e2	0%		0%		0%	0%	0%	0%	0%		0%		-0%	0%	0%	0%
14			e3	**-2%**		**2%**		0%	0%	0%	0%	**-2%**		-0%		**2%**	0%	0%	0%
15			e12	0%		0%		0%	0%	0%	0%	0%		0%		0%	0%	0%	0%
16			e13	1%		-1%		0%	0%	0%	0%	**2%**		**2%**		**2%**	0%	0%	0%
17			e23	0%		0%		0%	0%	0%	0%	0%		-0%		0%	0%	0%	0%
18			e123	0%		-0%		-0%	0%	0%	0%	-1%		-0%		0%	0%	0%	0%

Fig. 21. Noisy Math Function SRM 2^3 Cross Validation Results.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	BPD			Train						CI	CI	CI	Predict						CI	CI	CI
2	Nh	W,V	Gamma	90%	3SEM	95%	3SEM	99%	3SEM	90%	95%	99%	90%	3SEM	95%	3SEM	99%	3SEM	90%	95%	99%
3	-	-	-	89%	0.2%	93%	0.2%	98%	0.2%	6.2	7.4	9.8	90%	1.7%	93%	1.7%	99%	1.7%	6.2	7.4	9.8
4	+	-	-	89%	0.3%	94%	0.2%	99%	0.1%	5.7	6.8	8.9	**84%**	2.1%	**90%**	1.5%	**95%**	0.7%	5.7	6.8	8.9
5	-	+	-	89%	0.2%	93%	0.2%	98%	0.2%	6.2	7.4	9.7	88%	1.8%	93%	1.6%	98%	0.7%	6.2	7.4	9.7
6	+	+	-	89%	0.3%	94%	0.2%	99%	0.1%	5.7	6.8	8.9	**86%**	2.0%	**90%**	1.3%	**96%**	0.5%	5.7	6.8	8.9
7	-	-	+	93%	0.2%	96%	0.2%	99%	0.2%	6.4	7.6	10.0	89%	1.7%	**92%**	1.4%	**96%**	1.0%	6.4	7.6	10.0
8	+	-	+	**88%**	0.2%	94%	0.2%	99%	0.1%	6.0	7.1	9.4	**86%**	1.8%	**91%**	1.5%	98%	0.7%	6.0	7.1	9.4
9	-	+	+	93%	0.2%	96%	0.2%	99%	0.1%	6.4	7.6	10.0	89%	1.6%	**92%**	1.1%	**95%**	0.8%	6.4	7.6	10.0
10	+	+	+	88%	0.2%	94%	0.2%	99%	0.1%	6.0	7.1	9.3	**85%**	1.9%	**90%**	1.3%	98%	0.7%	6.0	7.1	9.3
11			e1	**-3%**		-0%		0%		**-0.4**	**-0.5**	**-0.7**	**-4%**		**-2%**		-0%		**-0.4**	**-0.5**	**-0.7**
12			e2	0%		0%		0%		0.0	0.0	0.0	-0%		-0%		-0%		0.0	0.0	0.0
13			e3	**2%**		1%		0%		**0.2**	**0.3**	**0.3**	0%		-0%		-0%		**0.2**	**0.3**	**0.3**
14			e12	0%		0%		0%		0.0	0.0	0.0	1%		0%		0%		0.0	0.0	0.0
15			e13	**-2%**		-1%		-0%		0.0	0.0	0.1	0%		1%		**3%**		0.0	0.0	0.1
16			e23	-0%		-0%		0%		0.0	0.0	0.0	-0%		-0%		-0%		0.0	0.0	0.0
17			e123	0%		-0%		0%		0.0	0.0	0.0	-1%		-0%		-0%		0.0	0.0	0.0

Fig. 22. Noisy Math Function BPD 2^3 Cross Validation Results.

BPFANN Cross Validation Findings

Table 3 summarizes our findings from Figures 19, 20, 21, and 22 for the training response, while Table 4 summarizes our findings for the predictive response.

Table. 3: 2^3 Factorial BPFANN Training Cross Validation Findings

Design Point	EDCM	SRM	BPD
(- - -)			
(+ - -)	low MSE/GTR		
(- + -)		low $\pm 2\sigma_{srm}$	
(+ + -)	low MSE/GTR	low $\pm 2\sigma_{srm}$	
(- - +)		low $\pm\sigma_{srm}$	
(+ - +)			low 90% BCI
(- + +)		low $\pm\sigma_{srm}$	
(+ + +)			

Table. 4: 2^3 Factorial BPFANN Predictive Cross Validation Findings

Design Point	EDCM	SRM	BPD
(- - -)			
(+ - -)	right skew/leptokurtosis	low $\pm(2,3)\sigma_{srm}$	low 90%,95%,99% BCI
(- + -)			
(+ + -)	right skew/leptokurtosis	low $\pm(2,3)\sigma_{srm}$	low 95%,99% BCI
(- - +)		low $\pm(1,2)\sigma_{srm}$	
(+ - +)			low 90%,95% BCI
(- + +)		low $\pm(1,2)\sigma_{srm}$	low 95%,99% BCI
(+ + +)			low 95% BCI

BPFANN Cross Validation Commentary

We provide commentary on both BPD and SRM predictive modeling for this case study.

SRM Data Summary

- Highest σ_{srm} responses closest to true value of σ_ϵ.
- Design point predictive SRM models within $\pm 3SEM$: (-,-,-),(-,+,-).
- Design point predictive SRM models marginally within $\pm 3SEM$: (+,-,+), (+,+,+).

- Design point predictive SRM models outside $\pm 3SEM$: (+,-,-),(+,+,-),(-,-,+),(-,+,+).

- SRM responses not affected by prior distribution setting for $\boldsymbol{\theta}$ as evident by factor effect **e2**.

BPD Data Summary

- Highest BCI widths (columns S,T,U Figure 22) are closest to correct values expected for Gaussian distribution for the true stochastic signature. We note that design point models (-,-,-), (-,+,-) are most correct to two significant figures (Section 6.1.1) and very close to the ground truth values we would expect.

- Design point predictive BPD models within $\pm 3SEM$: (-,-,-),(-,+,-).

- Design point predictive BPD models outside $\pm 3SEM$: (+,-,-),(+,+,-),(-,-,+),(-,+,+),(+,-,+),(+,+,+).

- BPD responses not affected by prior distribution setting for $\boldsymbol{\theta}$ as evident by factor effect **e2**.

Correspondence between Training and Predictive Responses

- There is a consistent pattern of negligible auto correlation for both training and predictive data regions (Figure 20) across all design points.

- Column E (Figure 19) is Pearson's R for the design point EDCM compared to the training data while column F is Pearson's R for the design point EDCM

compared to the true de-noised deterministic data. Column F shows a consistent pattern of better fits to the ground truth training deterministic data for the simpler networks (factor effect **e1**) indicating that the more complex networks are over-fitting. The same pattern is present for the prediction data although less severe.

- More complex networks consistently lower the MSE (Figure 19) for training as also shown by factor effect **e1**; however, **e1** also shows that Pearson's R for the fit of the EDCM to the true deterministic data also suffers and is an indication of over-fitting on both counts.

- For training we note little skew/kurtosis of concern as indicated by columns I,J.

- For the predictive data (Figure 19), fits to the true deterministic data are uniformly good except for the (+,+,+) design point which is conspicuously poor.

- For the predictive data, factor effects in Column M (Figure 19) rows 11-17 show a consistent pattern of poorer fits to the ground truth data as all factors go from their lower to higher settings.

- Several design points at the low end of predictive performance show MSE values close to ground truth for training but are noticeably larger for predictive data. By contrast, the best performing design points for prediction show consistent values of MSE from training to predictions and are also near the ground truth

value.

- *For predictions we note the onset of skew/kurtosis as indicated by columns P,Q, for the design points noted in training for suspiciously low MSE scores.*

Effect of Bayesian Prior on Regularization/Generalization

Figure 23 shows the effects of the Bayesian prior distribution on the values of the network parameters ω and ν as sampled by the Metropolis MCMC sampling procedure discussed in Section 4.1.6 and other measurements reported previously for the Noisy Math Function using the same split between training and predictive data used previously for all 200 patterns. Figures reported are for predictive data.

	A	B	C	D	E	F	G
1	Prior	(-3,-3,2)	(-2,-2,2)	(-1,-1,2)	(0,0,2)	(1,1,2)	(2,2,2)
2				EDCM			
3	MSE	11.7	4.9	4.21	5.4	5.81	5.58
4	R	0.21	0.71	0.72	0.75	0.83	0.78
5	Mean	−0.2	0.0	0.2	−0.1	0.2	−0.4
6	StdDev	3.06	2.07	2.08	2.24	2.28	2.29
7	Skew	−0.3	0.1	0.2	0.1	0.3	0.2
8	Kurt	0.4	−0.4	−0.2	0.3	0.0	−0.2
9				SRM			
10	1s	76%	62%	61%	61%	58%	60%
11	2s	95%	95%	93%	91%	91%	92%
12	3s	99%	99%	99%	99%	98%	98%
13				BPD			
14	90%	94%	89%	87%	83%	87%	86%
15	95%	95%	95%	93%	92%	92%	94%
16	99%	98%	99%	99%	97%	97%	97%
17	90% CI	11.7	6.79	6.45	6.68	6.75	7.02
18	95% CI	14	8.09	7.68	7.96	8.04	8.36
19	99% CI	18.3	10.6	10.1	10.5	10.6	11
20				MCMC			
21	sig_bl	3.6	2.07	1.97	2.04	2.06	2.14
22	W0	0.0043	0.044	0.37	4.4	45	467
23	Wi	0.0041	0.043	0.32	3.7	42	392
24	V	352	324	74	48	41	45

Fig. 23. Effects of Bayesian Prior on Regularization.

Row 1 of the above figure gives the value of the prior distribution and is encoded such that, for example, a prior designation of $(-2, -2, 2) \rightarrow \phi(0, 10^{-2}), \phi(0, 10^{-2}), \phi(0, 10^{2})$ as previously reported. Values for EDCM, SRM and BPD measurements for the predictive data in the same format used previously appear under each different prior setting. The MCMC section at the bottom reports the maximum absolute value of the indicated parameter observed in MCMC sampling where W_0 is for the hidden unit bias weights (ω_{0h}), W_i designates the hidden unit input weights (ω_{hi}), and V designates the hidden unit to output weights (ν_{ho}) of the deterministic model equation (14). Mean value for σ_{bl} observed in MCMC sampling is listed as "sig_bl".

These results show that for the Bayesian prior setting $(-1, -1, 2)$ (or $\phi(0, 10^{-1}), \phi(0, 10^{-1}), \phi(0, 10^{2})$) we find the following:

- lowest MSE,
- EDCM σ_{srm} value of 2.08 is closest to ground truth σ_{ϵ} value of 1.88,
- SRM measurements depart somewhat from ideal,
- BCI measurements close to ideal,
- BCI widths closest to ground truth values 62, 7.52, 9.68,
- MCMC sampling for σ_{bl} closest to ground truth value of 1.88,
- maximum observed magnitude for ω_0, ω_i, ν from MCMC sampling always at the same order of magnitude as the associated prior setting.

This last observation shows conclusively that the Bayesian prior is regulating as expected for all settings. It is important to note, however, that the expected associated generalization varies somewhat as the prior setting departs from this nominal value. An unexpected response was that BPFANN maintains close to nominal values for Bayesian predictive distributions over a broad range of prior settings. In certain extreme cases such as the prior (-3-3,2), BPFANN opts for larger values of σ_{bl} and scores a respectable but misleading predictive performance due to a poor EDCM fit. Figure (24) shows BCI predictive envelopes for the 90% BCI for priors (-3-3,2) to (0,0,2) in order left-to-right, top-to-bottom. Figure (25) is for priors (1,1,2) to (2,2,2). Figures (26) - (27) show histograms for the sampling of σ_{bl} for priors in the same order as the related BCI plots.

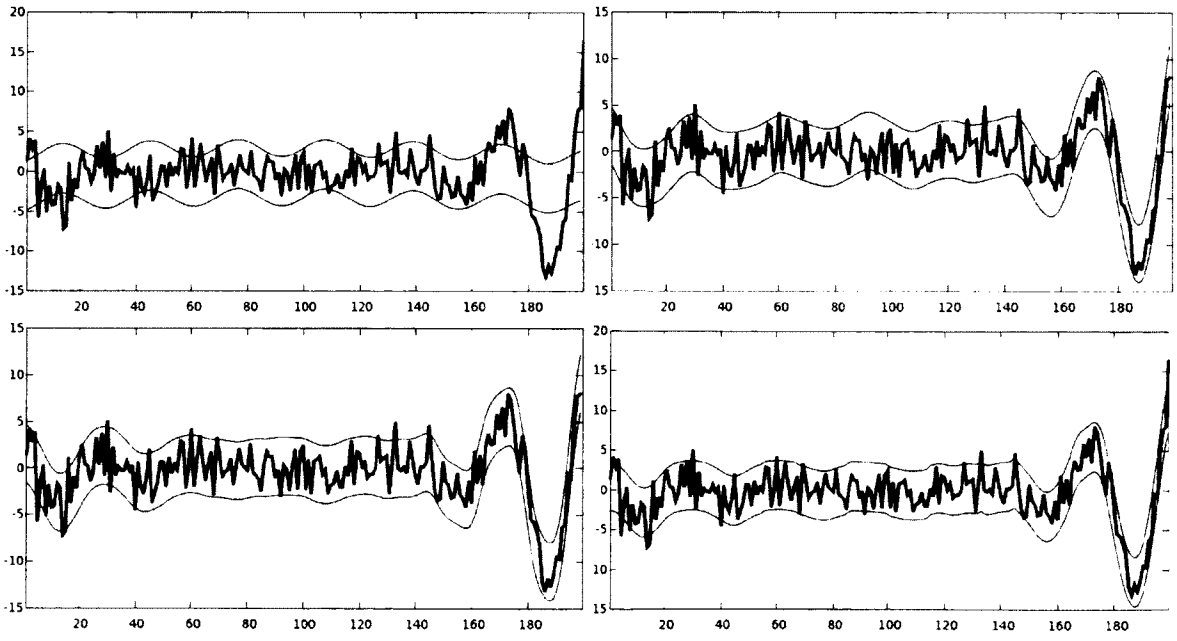


Fig. 24. 90% BCI Performance by Bayesian Prior.

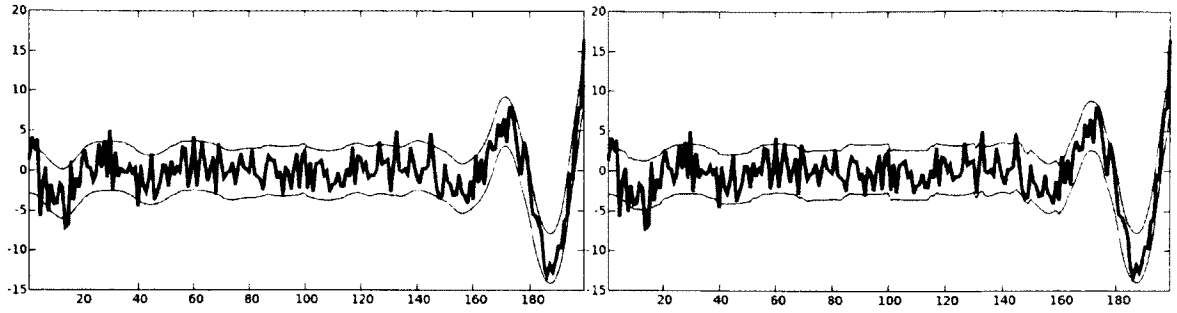


Fig. 25. 90% BCI Performance by Bayesian Prior.

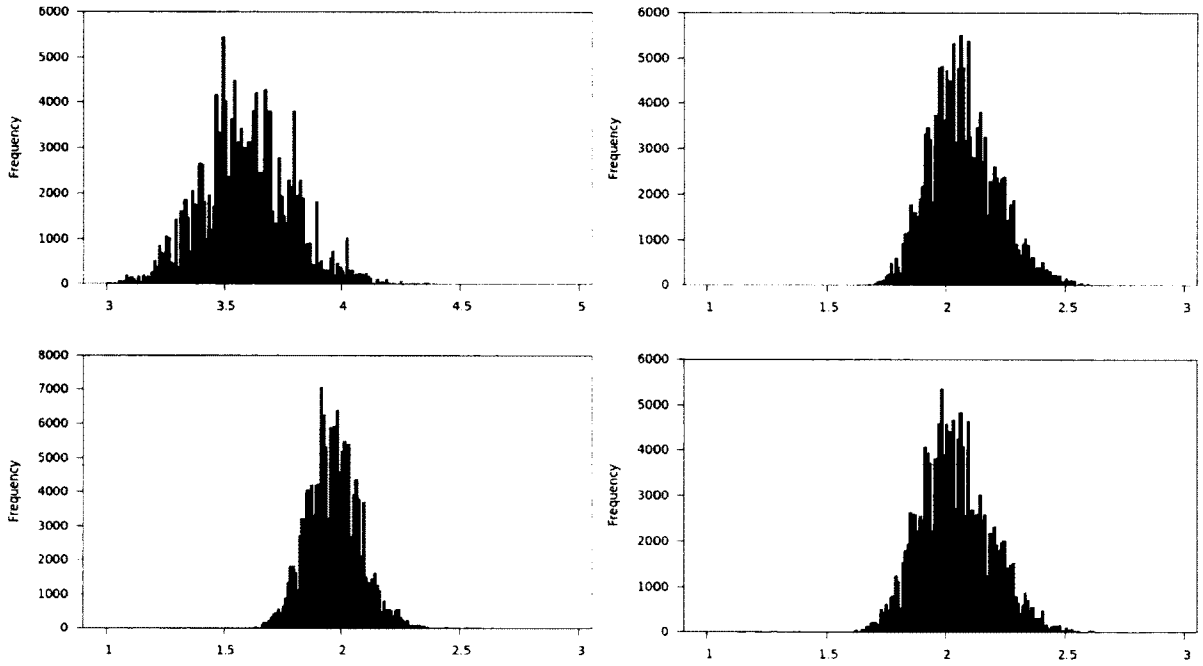
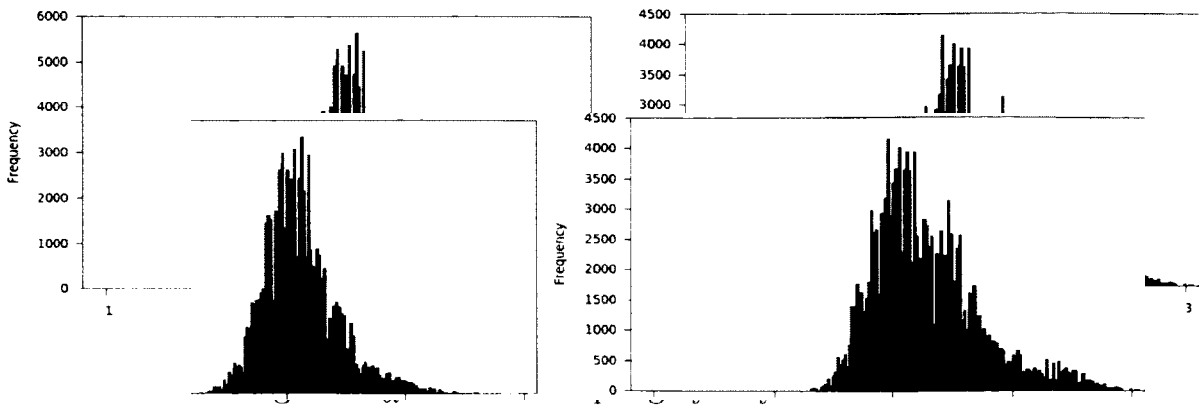


Fig. 26. σ_{bl} MCMC Sampling by Bayesian Prior.



From the above figures we note the following:

- BCI envelopes for prior (-1,-1,2) look best and track ground truth deterministic model best. Compare to Figure (18).
- Histogram for MCMC σ_{bl} sampling for prior (-1,-1,2) has best shape and appears to be best centered near the ground truth value of 1.88.
- The too narrow prior (-3,-3,2) samples for σ_{bl} in a suboptimal region far away from the correct value of 1.88
- The broader than optimal priors increasingly sample for σ_{bl} away from the optimal region.

Matlab Cross Validation Findings

Table 5 summarizes our findings from Figures 28 and 29 for training response. Table 6 summarizes our findings from Figures 28 and 29 for predictive response.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	SRM														
2	Train		+/- 0.01	+/- 0.001	Gauss	ACC									
3			MSE	R	GT R	1	2	3	4	5	6	7	8	9	10
4	-	-	3.98	0.435	0.725	−0.1	0.0	0.1	0.0	0.0	0.0	0.1	−0.1	0.1	0.0
5	+	-	3.68	0.498	0.876	−0.2	−0.1	0.0	0.0	0.0	0.0	0.1	−0.1	0.1	0.1
6	-	+	3.68	0.501	0.923	−0.2	−0.1	0.1	0.1	0.0	0.0	0.1	−0.1	0.1	0.0
7	+	+	3.66	0.504	0.926	−0.2	−0.1	0.0	0.0	0.0	0.0	0.1	−0.1	0.1	0.1
8		e1	−0.2	**0.03**	0.1	−0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9		e2	−0.2	**0.04**	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10		e12	0.1	−0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11															
12	Predict				Gauss	ACC									
13			MSE	R	GT R	1	2	3	4	5	6	7	8	9	10
14	-	-	3.32	0.841	0.895	0.0	0.0	−0.3	−0.1	−0.1	0.1	−0.1	−0.1	0.0	0.1
15	+	-	3.17	0.809	0.871	0.0	0.0	−0.3	0.0	0.0	0.2	−0.1	−0.2	−0.1	0.1
16	-	+	3.19	0.868	0.921	0.0	0.0	−0.4	0.0	0.0	0.1	−0.1	−0.2	−0.1	0.1
17	+	+	3.22	0.858	0.912	0.0	0.0	−0.4	−0.1	0.0	0.1	−0.1	−0.1	0.0	0.1
18		e1	−0.06	**−0.02**	−0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19		e2	−0.04	**0.04**	0.03	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20		e12	0.05	0.01	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig. 28. Noisy Math Function Matlab EDCM 2^2 Cross Validation Results.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	SRM						+/- 0.3%		+/- 0.2%				
2	Train					Gauss	68%	84%	96%	99.7%	100%	100%	100%
3			Mean	StDv	Skew	Kurt	1s	r2s	2s	3s	4s	5s	6s
4	-	-	0.03	1.99	0.00	0.00	69%	**81%**	95%	100%	100%	100%	100%
5	+	-	0.03	1.92	−0.14	−0.10	68%	**82%**	96%	100%	100%	100%	100%
6	-	+	0.02	1.92	−0.05	−0.04	**66%**	85%	96%	100%	100%	100%	100%
7	+	+	0.03	1.91	−0.04	0.04	68%	84%	96%	100%	100%	100%	100%
8		e1	0.01	−0.04	−0.07	−0.01	−0%	−0%	1%	−0%	0%	0%	0%
9		e2	0.00	−0.04	0.02	0.05	−1%	3%	1%	−0%	0%	0%	0%
10		e12	0.00	0.02	0.04	0.04	1%	−0%	−0%	0%	0%	0%	0%
11							+/- 0.3%		+/- 0.2%				
12	Predict						68%	84%	96%	99.7%	100%	100%	100%
13			Mean	StDv	Skew	Kurt	1s	r2s	2s	3s	4s	5s	6s
14	-	-	0.09	1.82	0.30	−0.08	70%	90%	97%	100%	100%	100%	100%
15	+	-	0.02	2.06	0.38	0.74	**64%**	86%	**93%**	99%	100%	100%	100%
16	-	+	0.08	2.04	0.48	−0.26	**63%**	84%	96%	100%	100%	100%	100%
17	+	+	0.04	1.89	0.38	−0.13	67%	86%	97%	100%	100%	100%	100%
18		e1	−0.06	0.04	−0.01	0.47	**−1%**	−1%	−1%	−1%	0%	0%	0%
19		e2	0.01	0.02	0.09	−0.53	**−2%**	**−3%**	1%	0%	0%	0%	0%
20		e12	0.01	**−0.10**	−0.04	−0.17	**2%**	1%	1%	0%	0%	0%	0%

Fig. 29. Noisy Math Function Matlab SRM 2^2 Cross Validation Results.

Table. 5: 2^2 Factorial Matlab Training Cross Validation Findings

Design Point	EDCM	SRM
(- -)	Highest MSE, lowest R,R_{gt}	low $\pm\sqrt{2}\sigma_{srm}$
(+ -)		low $\pm\sqrt{2}\sigma_{srm}$
(- +)		low $\pm\sigma_{srm}$
(+ +)		

Table. 6: 2^2 Factorial Matlab Predictive Cross Validation Findings

Design Point	EDCM	SRM
(- -)		
(+ -)		low $\pm 1, \pm 2\sigma_{srm}$
(- +)		low $\pm 2\sigma_{srm}$
(+ +)		low $\pm\sigma_{srm}$

Matlab SRM Cross Validation Commentary

- Design point predictive SRM models within $\pm 3SEM$: (+,+).
- Design point predictive SRM models outside $\pm 3SEM$: (-,-), (,+),(+,-).
- MacKay's Bayesian Regression (MBR) method appears to give good generalization only for network with larger complexity.

Visual Comparison of BPFANN to Matlab

Comparisons of both highest and lowest predictive performers in terms of the design point choices for both BPFANN and Matlab example *composite* EDCM models to the actual ground truth noise-free deterministic data are provided in Figures 30, 31, 32, 33, 34, and 35. The Matlab composite models are a simple average of all models produced during training. The BPFANN composite models are produced from the total MCMC chain produced during training.

In the BPFANN diagrams, Figures 30 and 31, the smooth heavier line is the ground truth deterministic model (compare to Figure 18); the other relatively smooth line is the composite EDCM; the light noisy line is given data; the dotted enveloping lines are the SRM $\pm 2\sigma_{srm}$ lines, and the solid enveloping lines are the 95% BCI lines.

In the Matlab diagrams, the smooth heavier line is the ground truth deterministic model (compare to Figure 18). The other relatively smooth line is the composite EDCM, the light noisy line is given data, the solid enveloping lines are the SRM $\pm 2\sigma_{srm}$ lines.

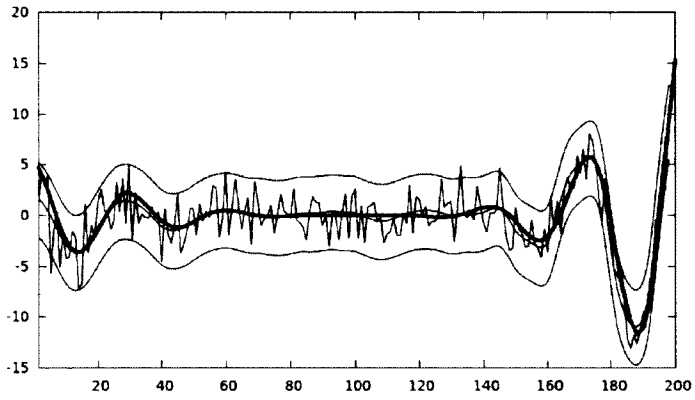


Fig. 30. Noisy Math Function BPD/SRM (-,-,-) Ground Truth Comparison.

The best performing predictive BPFANN design point (-,-,-) composite model in Figure 30 appears on the whole to be performing very well.

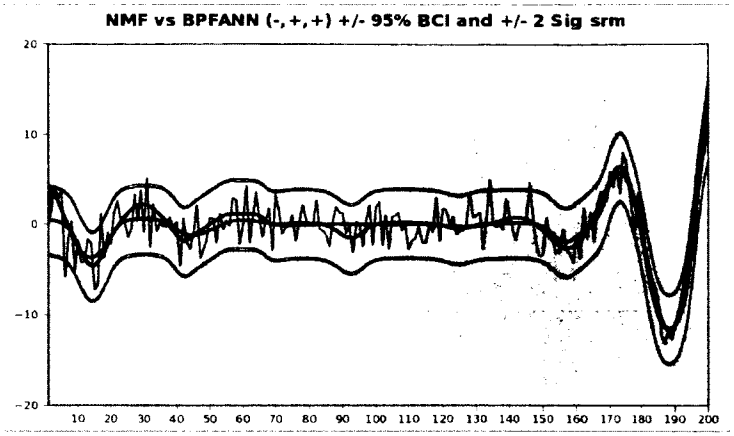


Fig. 31. Noisy Math Function BPD/SRM (-,+,+) Ground Truth Comparison.

The worst performing predictive BPFANN design point (-,+,+) composite model in Figure 31 appears on the whole to be slightly over-fitting in some regions and slightly under-fitting in others.

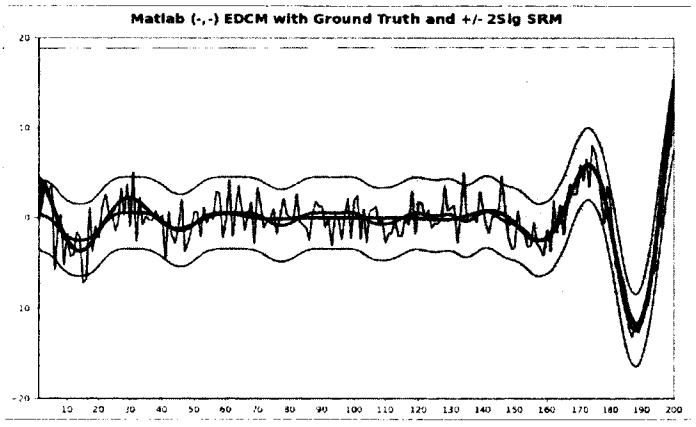


Fig. 32. Noisy Math Function Matlab SRM (-,-) Ground Truth Comparison.

The best performing predictive Matlab design point (-,-) composite model in Figure 32 appears on the whole to be slightly over-fitting in some regions and slightly under-fitting in others.

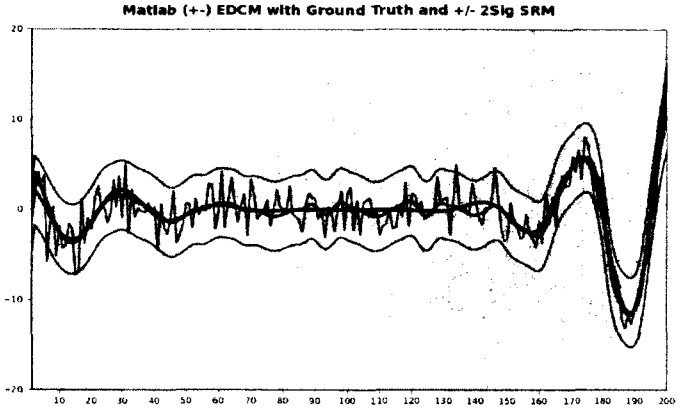


Fig. 33. Noisy Math Function Matlab SRM (+-) Ground Truth Comparison.

The worst performing predictive Matlab design point (+,-) composite model in Figure 33 appears on the whole to be slightly over-fitting.

Stochastic Residual Figure 34 is a comparison of histograms of the residual data for the entire set of trials in the cross validations for best performing design points for both BPFANN and Matlab, while Figure 35 is a comparison of histograms of the residual data for worst performing design points for each. The values listed for μ_{srm}, σ_{srm}, γ_{srm}, and κ_{srm} in Figures 19 and 29 are derived from these distributions but computed by the procedure described in section 5.2.

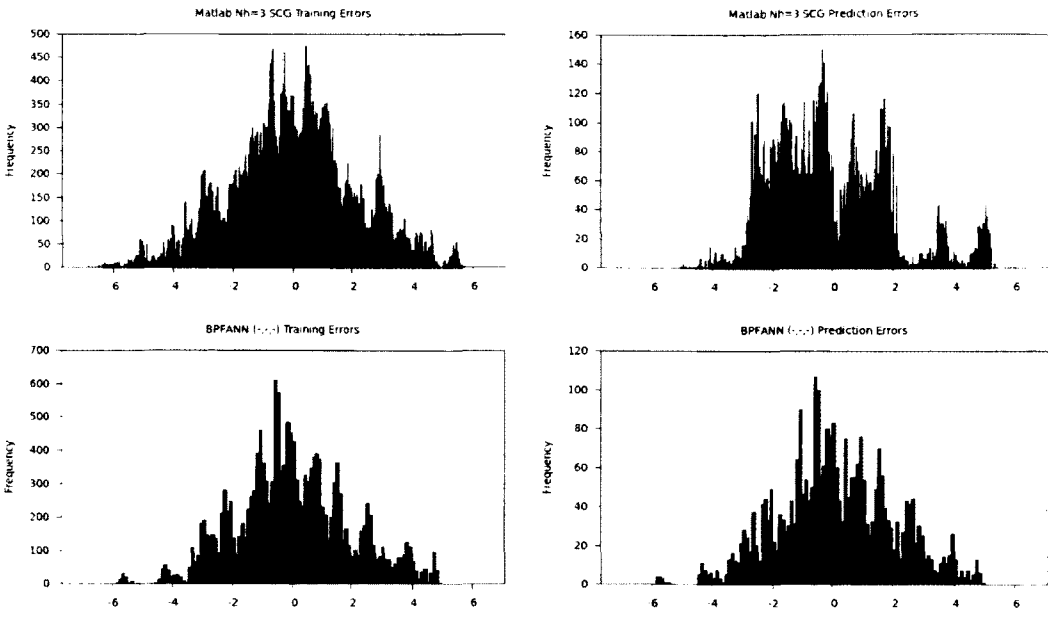


Fig. 34. Noisy Math Function Matlab BPFANN Residual Comparison.

Histograms of the stochastic residuals over all trials in the selected cross validations in Figure 34 for the best predictive performers show a departure from the expected Gaussian shape for Matlab for the predictive data; otherwise, all others are approximately Gaussian with approximately the correct ground truth standard deviation value of 1.88.

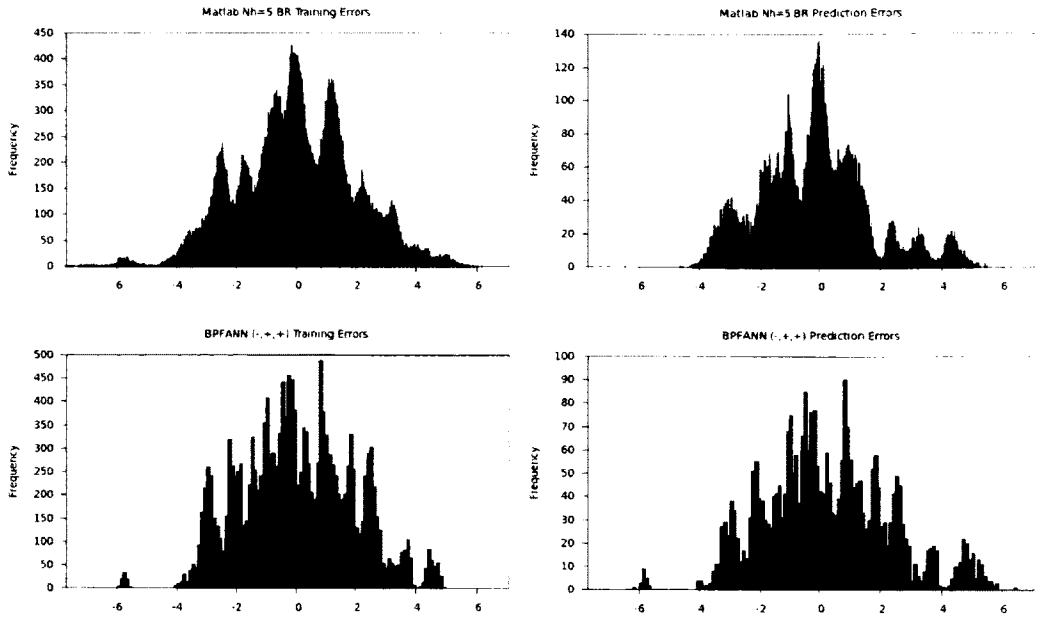


Fig. 35. Noisy Math Function Matlab BPFANN Residual Comparison.

Histograms of the stochastic residuals over the entire set of cross validation trials in Figures 35 for the worst predictive performers are slightly less Gaussian shaped than their best performer counterparts, but they are still near the correct ground truth standard deviation value of 1.88.

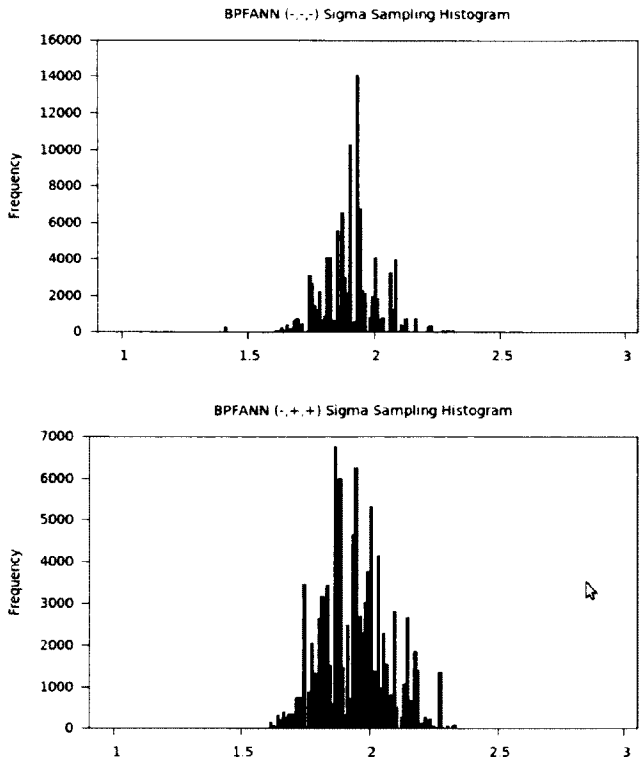


Fig. 36. Noisy Math Function BPFANN σ_{bl} Sampling.

Histograms of the σ_{bl} sampling for the BPFANN design point models (-,-,-) and (-,+,+) derived from their respective MCMC chains in Figure 36 indicate sampling centered on approximately the correct ground truth value of 1.88 in both cases, with the lower predictive performance model (-,+,+) showing a slightly greater spread in the sampling. The values listed for σ_{bl} in Figure 22 are derived from these distributions but are computed by the procedure described in section 5.2 using MCMC estimate of the marginalized posterior distribution equation (25).

6.2 CASE STUDY II NON-LINEAR REGRESSION

In this case study we analyze a data set[58] that provides a measure of the workability of high strength concrete[59]. The data set includes 103 data points. There are 8 input variables and 3 output variables in the data set. Here we model only the first output (the harder of the three) for predictive test purposes.

Input Attribute Information Input variables (component kg in one M^3 concrete):

- Sequence Number
- Cement
- Slag
- Fly ash
- Water
- Super-Plasticizer
- Coarse Aggregate
- Fine Aggregate

Output Attribute Information Slump Flow (cm). Raw data for the output Slump Flow is depicted in Figure 37.

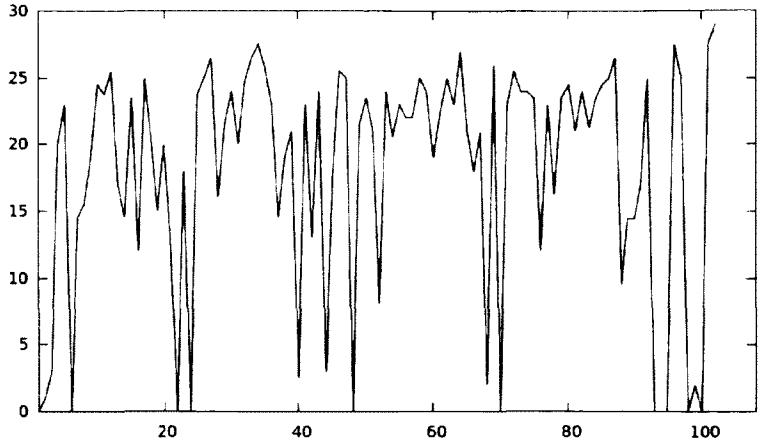


Fig. 37. Concrete Slump.

6.2.1 CROSS VALIDATION

We perform a cross validation study on this data for 1000 trials for both BPFANN (BPD/SRM) and Matlab (SRM) with the parameter setting matrices for both the 2^3 and 2^2 cross validations as listed in Tables 7 and 8 as per sections 5.4.2 and 5.4.1. Colored items stand out for particular commentary.

Table. 7: 2^3 Factorial Bayesian MCMC Design Point Settings.

Design Point	N_h	$P_0(\boldsymbol{\theta})$	$P_0(\sigma_{bl}) = \Gamma(shape, scale)$
(- - -)	4	$\phi(0,1), \phi(0,10^{-1}), \phi(0,10)$	$\Gamma(2,1)$
(+ - -)	8	$\phi(0,1), \phi(0,10^{-1}), \phi(0,10)$	$\Gamma(2,1)$
(- + -)	4	$\phi(0,100), \phi(0,100), \phi(0,100)$	$\Gamma(2,1)$
(+ + -)	8	$\phi(0,100), \phi(0,100), \phi(0,100)$	$\Gamma(2,1)$
(- - +)	4	$\phi(0,1), \phi(0,10^{-1}), \phi(0,10)$	$\Gamma(2,2)$
(+ - +)	8	$\phi(0,1), \phi(0,10^{-1}), \phi(0,10)$	$\Gamma(2,2)$
(- + +)	4	$\phi(0,100), \phi(0,100), \phi(0,100)$	$\Gamma(2,2)$
(+ + +)	8	$\phi(0,100), \phi(0,100), \phi(0,100)$	$\Gamma(2,2)$

Table. 8: 2^2 Factorial Matlab Design Point Settings.

Design Point	N_h	Training Method
(- -)	4	SCG
(+ -)	8	SCG
(- +)	4	MBR
(+ +)	8	MBR

Cross Validation Responses

The SRM and BPD 2^3 cross validation results for BPFANN are in Figures 38, 39, 40, and 41, and the results for Matlab SRM in Figures 42 and 43.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Train									Predict					
2	Nh	W,V	Gamma	MSE	R	Mean	StDv	Skew	Kurt	MSE	R	Mean	StDv	Skew	Kurt
3	-	-	-	29.2	0.788	−0.38	5.39	−1.1	3.4	32.3	0.724	0.10	5.27	−0.6	0.5
4	+	-	-	12.9	0.912	0.01	3.58	0.1	1.0	20.7	0.836	0.44	4.36	0.3	0.2
5	-	+	-	29.7	0.787	−0.35	5.38	−1.1	3.2	29.8	0.714	−0.13	5.4	−0.6	0.5
6	+	+	-	12.7	0.912	−0.02	3.57	0.1	1.1	19.6	0.861	0.63	4.28	0.4	0.2
7	-	-	+	29.9	0.783	−0.37	5.46	−1.1	3.2	32.2	0.716	0.17	5.2	−0.4	0.1
8	+	-	+	12.8	0.912	0.00	3.57	0.1	1.0	20.2	0.85	0.46	4.21	0.3	0.2
9	-	+	+	29.2	0.787	−0.37	5.39	−1.2	3.4	30.8	0.729	0.02	5.26	−0.5	0.2
10	+	+	+	12.8	0.912	−0.02	3.57	0.1	1.1	19.1	0.846	0.59	4.22	0.4	0.2
11			e1	−16.7	0.13	0.36	−1.83	1.25	−2.3	−11.4	0.13	0.5	−1.0	0.9	−0.1
12			e2	−0.1	0.00	0.00	−0.02	−0.02	0.03	−1.5	0.01	0.0	0.0	0.0	0.0
13			e3	0.0	0.00	−0.01	0.02	0.00	0.01	0.0	0.00	0.0	−0.1	0.1	−0.2
14			e12	0.0	0.00	−0.02	0.02	0.00	0.00	0.4	0.00	0.2	−0.1	0.0	0.0
15			e13	−0.1	0.00	0.00	−0.02	−0.01	−0.02	−0.5	0.00	−0.1	0.0	−0.1	0.1
16			e23	−0.2	0.00	0.00	−0.01	−0.01	0.10	0.3	0.00	0.0	0.0	0.0	0.0
17			e123	0.4	0.00	0.01	0.02	0.01	−0.09	−0.3	−0.01	0.0	0.0	0.0	0.0

Fig. 38. Concrete Slump Flow EDCM 2^3 Cross Validation Results.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Train			ACC									
2	Nh	W,V	Gamma	1	2	3	4	5	6	7	8	9	10
3	-	-	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	+	-	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	-	+	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	+	+	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	-	-	+	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	+	-	+	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	-	+	+	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	+	+	+	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11			e1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12			e2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13			e3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14			e12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15			e13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
16			e23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17			e123	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18													
19	Predict			ACC									
20	Nh	W,V	Gamma	1	2	3	4	5	6	7	8	9	10
21	-	-	-	0.0	−0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	+	-	-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23	-	+	-	0.0	0.0	0.0	0.0	0.0	−0.1	0.0	0.0	0.0	−0.1
24	+	+	-	0.1	0.0	−0.1	0.0	0.0	0.0	0.0	0.0	−0.1	0.0
25	-	-	+	0.0	0.0	0.0	0.0	0.0	0.0	−0.1	0.0	−0.1	0.0
26	+	-	+	0.0	0.0	0.0	0.0	−0.1	0.0	0.0	0.1	0.1	0.0
27	-	+	+	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
28	+	+	+	0.0	0.0	0.0	−0.1	0.0	0.0	0.0	0.1	0.0	0.0
29			e1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30			e2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31			e3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
32			e12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
33			e13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
34			e23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35			e123	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig. 39. Concrete Slump Flow ACC 2^3 Cross Validation Results.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1				Train								Predict							
2				68%		96%		99.7%	100%	100%	100%	68%		96%		99.7%	100%	100%	100%
3	Nh	W,V	Gamma	1s	3SEM	2s	3SEM	3s	4s	5s	6s	1s	3SEM	2s	3SEM	3s	4s	5s	6s
4	-	-	-	75%	1.1%	96%	0.3%	**98%**	99%	100%	100%	70%	3.4%	96%	0.3%	98%	99%	100%	100%
5	+	-	-	69%	1.2%	95%	0.2%	**98%**	100%	100%	100%	**61%**	3.4%	**85%**	1.9%	**96%**	100%	100%	100%
6	-	+	-	75%	1.3%	96%	0.2%	**98%**	99%	100%	100%	69%	3.4%	96%	1.9%	98%	98%	100%	100%
7	+	+	-	69%	1.4%	95%	0.2%	**98%**	100%	100%	100%	**61%**	3.7%	**87%**	1.7%	**96%**	100%	100%	100%
8	-	-	+	75%	1.3%	96%	0.2%	**98%**	99%	100%	100%	70%	3.4%	96%	1.9%	98%	99%	100%	100%
9	+	-	+	69%	1.3%	95%	0.2%	**98%**	100%	100%	100%	**61%**	3.7%	**87%**	2.0%	**97%**	100%	100%	100%
10	-	+	+	76%	1.3%	96%	0.2%	**98%**	99%	100%	100%	69%	3.4%	96%	1.9%	98%	99%	100%	100%
11	+	+	+	70%	1.3%	95%	0.2%	**98%**	100%	100%	100%	**62%**	3.7%	**87%**	1.8%	**96%**	100%	100%	100%
12			e1	-6%		-2%		0%	1%	0%	0%	-8%		-10%		-2%	1%	0%	0%
13			e2	0%		-0%		-0%	-0%	0%	0%	0%		0%		0%	-0%	0%	0%
14			e3	0%		0%		-0%	-0%	0%	0%	0%		0%		0%	0%	-0%	0%
15			e12	-0%		-0%		-0%	-0%	0%	0%	1%		1%		0%	0%	0%	0%
16			e13	-0%		-0%		0%	0%	0%	0%	-0%		0%		-0%	0%	0%	0%
17			e23	1%		0%		0%	-0%	0%	0%	0%		-1%		-0%	0%	0%	0%
18			e123	-0%		-0%		0%	0%	0%	0%	0%		-0%		-0%	-0%	-0%	0%

Fig. 40. Concrete Slump Flow SRM 2^3 Cross Validation Results.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	BPD			Train						CI	CI	CI	Predict						CI	CI	CI
2	Nh	W,V	Gamma	90%	3SEM	95%	3SEM	99%	3SEM	90%	95%	99%	90%	3SEM	95%	3SEM	99%	3SEM	90%	95%	99%
3	-	-	-	93%	0.3%	96%	0.2%	98%	0.2%	15.4	18.4	24.1	89%	2.4%	96%	2.0%	97%	1.9%	15.4	18.4	24.1
4	+	-	-	91%	0.3%	94%	0.2%	97%	0.2%	11.7	13.9	18.3	**82%**	2.4%	**85%**	2.0%	**90%**	1.8%	11.7	13.9	18.3
5	-	+	-	93%	0.3%	96%	0.2%	98%	0.2%	15.4	18.4	24.1	90%	2.4%	97%	2.0%	97%	1.9%	15.4	18.4	24.1
6	+	+	-	91%	0.3%	94%	0.2%	97%	0.2%	11.7	13.9	18.3	**84%**	2.5%	**86%**	1.9%	**90%**	1.8%	11.7	13.9	18.3
7	-	-	+	94%	0.3%	97%	0.2%	98%	0.2%	16.2	19.3	25.4	93%	2.4%	97%	2.0%	98%	1.9%	16.3	19.4	25.4
8	+	-	+	91%	0.3%	94%	0.2%	97%	0.2%	12.0	14.3	18.7	**84%**	2.0%	**87%**	1.9%	**93%**	2.2%	12.0	14.3	18.7
9	-	+	+	94%	0.3%	97%	0.2%	98%	0.2%	16.2	19.2	25.3	92%	2.4%	96%	1.9%	98%	1.9%	16.2	19.3	25.3
10	+	+	+	92%	0.3%	95%	0.3%	97%	0.1%	11.9	14.2	18.7	**84%**	2.0%	**86%**	1.9%	**92%**	1.9%	11.9	14.2	18.7
11			e1	-2%		-2%		-1%		-4.0	-4.7	-6.2	-8%		-10%		-7%		-4.0	-4.8	-6.2
12			e2	-0%		0%		0%		0.0	-0.1	0.0	0%		-0%		-0%		-0.9	-1.0	-1.4
13			e3	1%		0%		0%		0.5	0.6	0.8	2%		1%		2%		-2.3	-2.7	-3.6
14			e12	0%		0%		0%		0.0	0.0	0.0	0%		0%		-0%		-2.1	-2.6	-3.4
15			e13	-1%		0%		-0%		-0.3	-0.3	-0.4	-1%		1%		1%		-1.3	-1.6	-2.1
16			e23	0%		0%		0%		0.0	-0.1	0.0	-1%		-1%		-0%		-3.1	-3.7	-4.8
17			e123	0%		0%		-0%		0.0	0.0	0.0	-0%		-0%		-0%		-0.7	-0.8	-1.1

Fig. 41. Concrete Slump Flow BPD 2^3 Cross Validation Results.

BPFANN Cross Validation Findings

Figure 38 shows a consistent pattern of lowest MSE, highest R, lowest σ_{srm}, lowest residual γ_{srm}, and κ_{srm} for the more complex networks as opposed to the lowest, as clearly captured by the factor effect **e1** for both training and prediction. Presence of substantial excess skew and kurtosis in the training response translates

much less to the prediction response. From Figure 39, we note little residual autocorrelation for both training and prediction. In Figure 40 we note that all networks report good figures for training whereas the less complex networks have consistently acceptable figures for Gaussian inclusion percentages based on σ_{srm} for predictions, and the more complex networks are off the mark. Figure 41 shows the same pattern as for SRM data with the less complex networks giving consistently acceptable prediction figures based on BCIs.

Matlab Cross Validation Findings

Figure 42 shows a consistent pattern of lowest MSE, and lowest σ_{srm} for training whereas the R values are remarkably consistent. Also for training, all design point except (-,-) show considerable excess kurtosis. Gaussian SRM inclusion percentages are consistently acceptable for training with the exception of the (+,+) design point which is marginally off on one of the measures.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	SRM													
2	Train		+/- 0.5	+/- 0.004	ACC									
3			MSE	R	1	2	3	4	5	6	7	8	9	10
4	-	-	17	0.862	−0.1	0.0	0.1	0.0	0.0	0.0	0.1	−0.1	0.1	0.0
5	+	-	15.5	0.878	−0.2	−0.1	0.0	0.0	0.0	0.0	0.1	−0.1	0.1	0.1
6	-	+	14.4	0.885	−0.2	−0.1	0.1	0.1	0.0	0.0	0.1	−0.1	0.1	0.0
7	+	+	12	0.903	−0.2	−0.1	0.0	0.0	0.0	0.0	0.1	−0.1	0.1	0.1
8		e1	−2.0	0.02	−0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9		e2	−3.0	0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10		e12	−0.2	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11														
12	Predict		+/- 0.7		ACC									
13			MSE	R	1	2	3	4	5	6	7	8	9	10
14	-	-	13.1	0.952	0.0	0.0	−0.3	−0.1	−0.1	0.1	−0.1	−0.1	0.0	0.1
15	+	-	12.9	0.951	0.0	0.0	−0.3	0.0	0.0	0.2	−0.1	−0.2	−0.1	0.1
16	-	+	10.5	0.962	0.0	0.0	−0.4	0.0	0.0	0.1	−0.1	−0.2	−0.1	0.1
17	+	+	8.31	0.97	0.0	0.0	−0.4	−0.1	0.0	0.1	−0.1	−0.1	0.0	0.1
18		e1	−1.19	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19		e2	−3.59	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20		e12	−0.50	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig. 42. Concrete Slump Flow Matlab EDCM 2^2 Cross Validation Results.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	SRM						+/- 0.4%		+/- 0.2%				
2	Train					Gauss	68%	84%	96%	99.7%	100%	100%	100%
3			Mean	StDv	Skew	Kurt	1s	r2s	2s	3s	4s	5s	6s
4	-	-	−0.10	4.19	−0.28	0.39	72%	86%	95%	100%	100%	100%	100%
5	+	-	−0.25	3.94	−0.80	1.77	70%	88%	96%	99%	100%	100%	100%
6	-	+	−0.07	3.81	−0.52	2.44	74%	87%	95%	98%	100%	100%	100%
7	+	+	−0.08	3.54	−0.45	8.53	84%	90%	**94%**	97%	99%	100%	100%
8		e1	−0.08	**−0.26**	**−0.22**	**3.73**	**4%**	2%	−0%	−1%	−1%	0%	0%
9		e2	0.10	**−0.39**	**0.06**	**4.40**	**8%**	2%	−1%	−2%	−1%	0%	0%
10		e12	0.04	0.00	**0.15**	**1.18**	**3%**	0%	−1%	0%	−0%	0%	0%
11							+/- 0.3%		+/- 0.2%				
12	Predict						68%	84%	96%	99.7%	100%	100%	100%
13			Mean	StDv	Skew	Kurt	1s	r2s	2s	3s	4s	5s	6s
14	-	-	0.95	3.49	0.76	0.45	81%	93%	94%	100%	100%	100%	100%
15	+	-	0.65	3.54	−0.10	−0.22	73%	81%	98%	100%	100%	100%	100%
16	-	+	0.50	3.09	0.50	0.62	80%	88%	96%	99%	100%	100%	100%
17	+	+	0.21	2.68	0.42	1.16	84%	92%	96%	99%	99%	100%	100%
18		e1	**−0.29**	**−0.18**	**−0.47**	**−0.07**	**−2%**	**−4%**	2%	−0%	−1%	0%	0%
19		e2	**−0.44**	**−0.63**	**0.13**	**0.78**	**5%**	**3%**	0%	−1%	−1%	0%	0%
20		e12	0.00	−0.11	0.20	0.30	3%	4%	−1%	−0%	−0%	0%	0%

Fig. 43. Concrete Slump Flow Matlab SRM 2^2 Cross Validation Results.

From Figure 42, we note little residual auto-correlation for both training and prediction although slightly more than for BPFANN. Prediction responses show a

mixed pattern of moment responses though SRM inclusion percentages are consistently good across all design points.

Visual Comparison of BPFANN to Matlab

Comparisons of both highest and lowest predictive performers in terms of the design point choices for both BPFANN and Matlab example *composite* EDCM models are provided in Figures 44, 45, 46, and 47. The Matlab composite models are a simple average of all models produced during training. The BPFANN composite models are produced from the total MCMC chain produced during training. In the following figures, the heavy line is the given data, the long dashed central line is the EDCM, the outer enveloping dotted lines are the SRM $\pm 2\sigma_{srm}$ lines, and the outer enveloping short dashed lines are the BPD $\pm$95% BCI lines.

The worst performing predictive BPFANN design point (-,-,-) composite model in Figure 44 appears on the whole to be potentially *over-regularizing* as it is an open question whether the flat sections in the plot are actually noise or part of the underlying deterministic data not modeled by the associated EDCM. We also note that the BCI prediction envelope is consistently inside the counterpart SRM envelope.

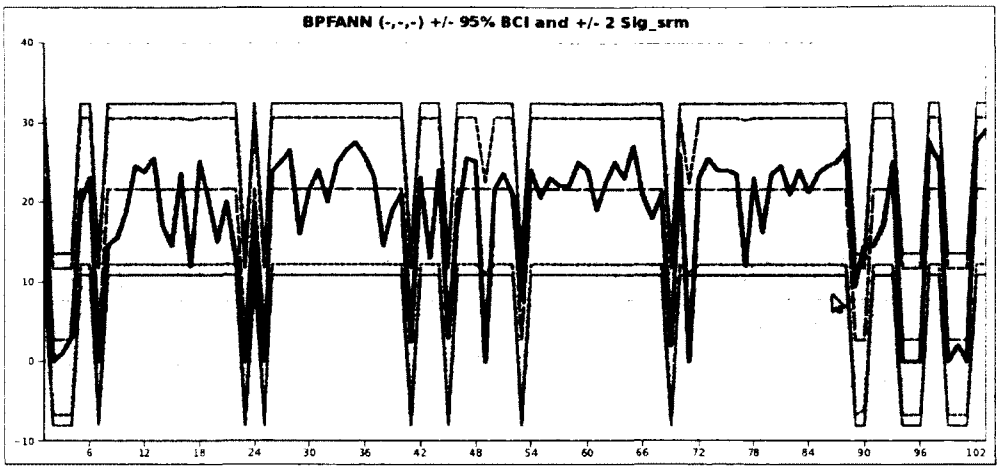


Fig. 44. Concrete Slump Flow BPD/SRM (-,-,-).

The best performing predictive BPFANN design point (+,+,+) composite model in Figure 45 appears on the whole to give an EDCM which is following the given data fairly closely. Here we also note that the BCI prediction envelope is almost identical with the counterpart SRM envelope.

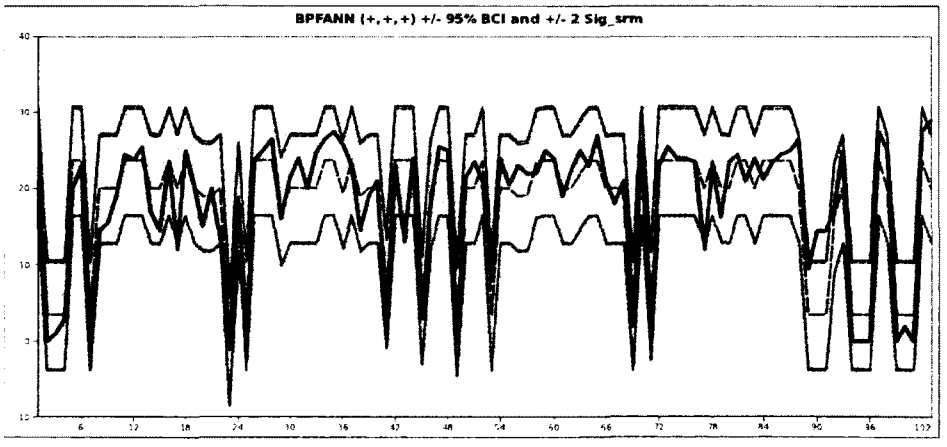


Fig. 45. Concrete Slump Flow BPD/SRM (+,+,+).

The worst performing predictive Matlab design point (-,-) composite model in Figure

46 appears on the whole to closely follow the given data with its EDCM except in a few places where there are a few notable departures.

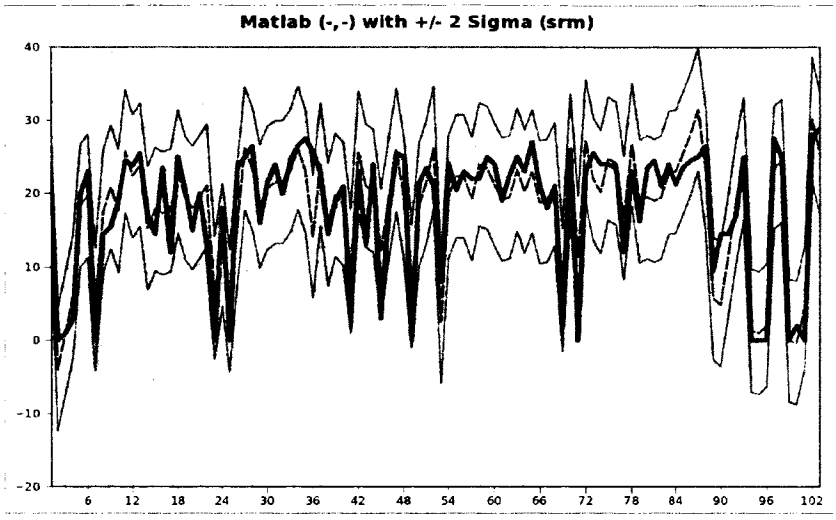


Fig. 46. Concrete Slump Flow Matlab SRM (-,-).

The best performing predictive Matlab design point (+,+) composite model in Figure 47 appears on the whole to closely follow the given data with its EDCM.

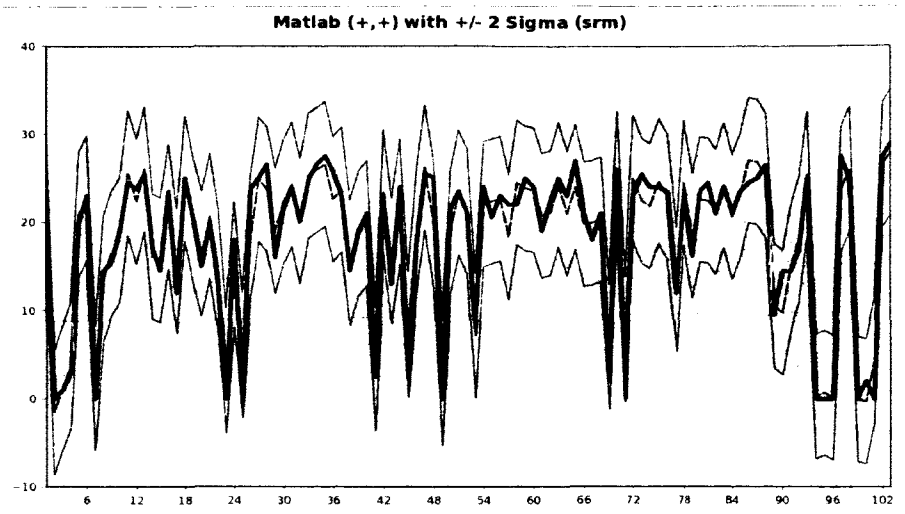


Fig. 47. Concrete Slump Flow Matlab SRM (+,+).

Stochastic Residual Figures 48 and 49 are a comparison of histograms of the residual data for each.

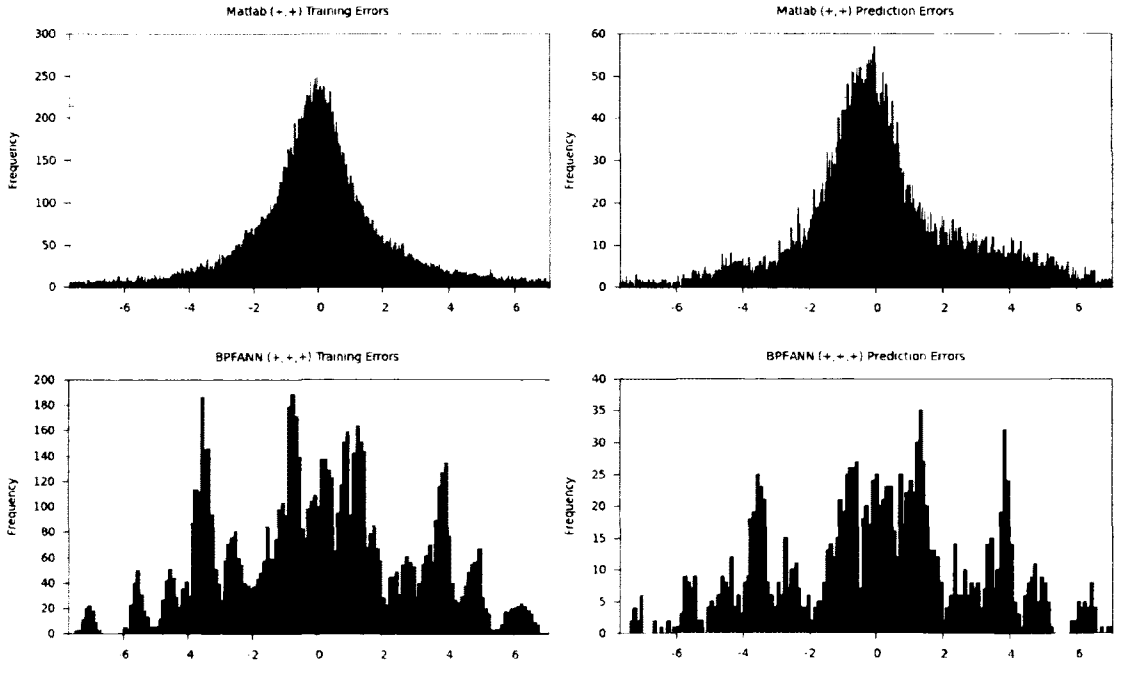


Fig. 48. Concrete Slump Flow Matlab BPFANN Best Performer Residuals.

Histograms of the stochastic residuals over all trials in the selected cross validations show a consistent pattern of superior Gaussian shape for the training data and are approximately the same size/shape over the predictive data portion.

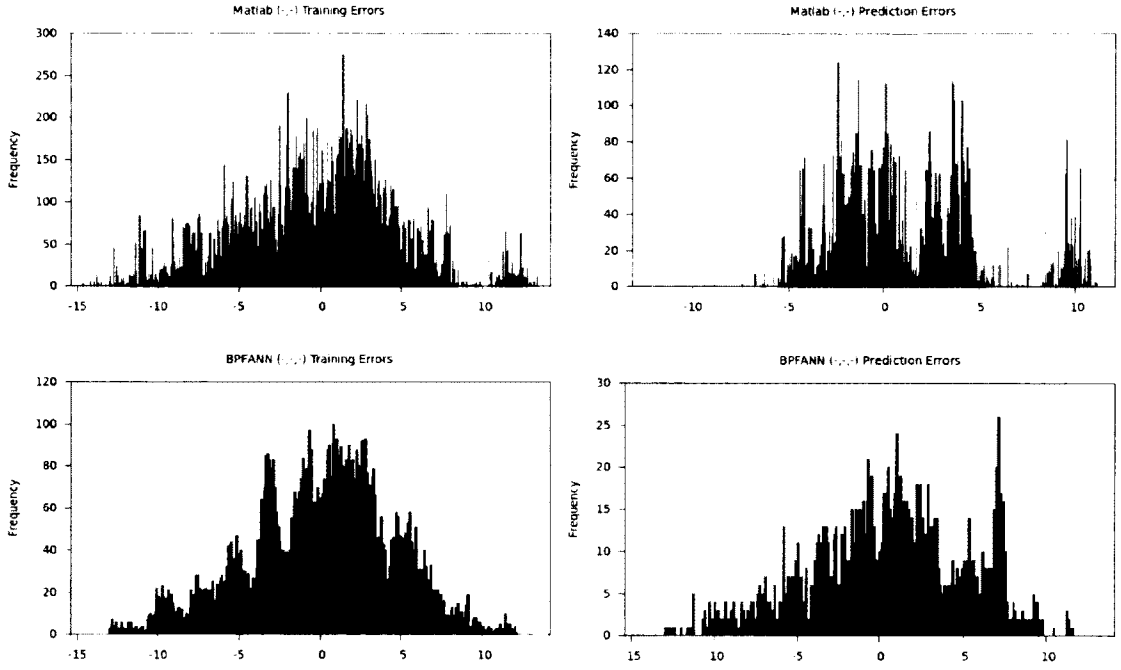


Fig. 49. Concrete Slump Flow Matlab BPFANN Worst Performer Residuals.

Histograms of the σ_{bl} sampling for the BPFANN design point models (-,-,-) and (-,+,+) derived from their respective MCMC chains in Figure 50 indicate a narrower and more Gaussian-like shape for the best predictive performer, with a mean value for σ_{bl} similar to that obtained for σ_{srm} by Matlab.

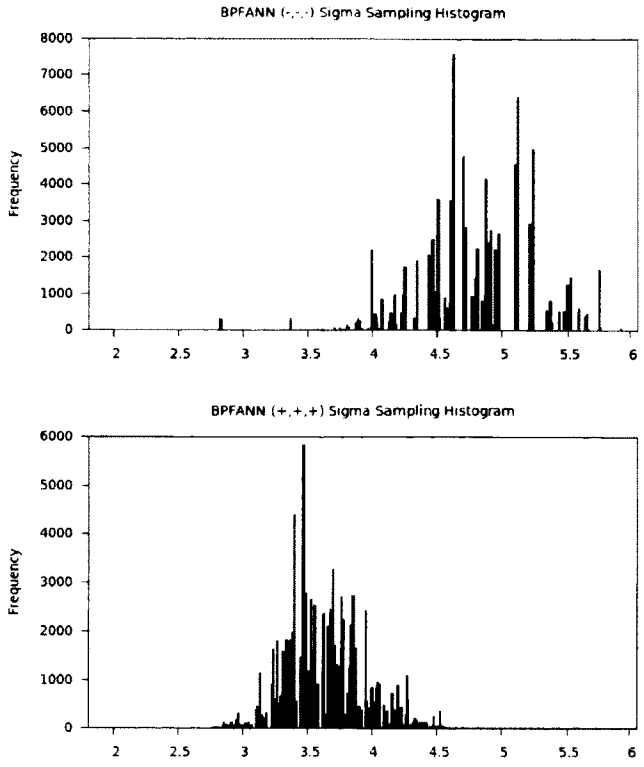


Fig. 50. Concrete Slump Flow BPFANN σ_{bl} Sampling.

6.3 CASE STUDY III CLASSIFICATION

In this case study we analyze one of the best known data sets in the pattern recognition literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other two; the latter two are not linearly separable from each other.

Input Attributes The input attributes for each pattern are:

- Sepal length in cm
- Sepal width in cm
- Petal length in cm

- Petal width in cm

Output Attributes Output Classification is a one-of-three binary encoded selection vector (see section 4.2.2) for the possibilities:

- Iris Setosa
- Iris Versicolour
- Iris Virginica

6.3.1 CROSS VALIDATION

We perform a cross validation study on this data for 1000 trials for both BPFANN (BPD/SRM) and Matlab (SRM) with the parameter setting matrices for both the 2^3 and 2^2 cross validations as listed in Tables 9 and 10 as per sections 5.4.2 and 5.4.1.

Table. 9: 2^2 Iris BPFANN Design Point Settings

Design Point	N_h	$P_0(\boldsymbol{\theta})$
(- -)	1	$\phi(0,1), \phi(0,1), \phi(0,1)$
(+ -)	3	$\phi(0,1), \phi(0,1), \phi(0,1)$
(- +)	1	$\phi(0,100), \phi(0,100), \phi(0,100)$
(+ +)	3	$\phi(0,100), \phi(0,100), \phi(0,100)$

Table. 10: 2^2 Iris Factorial Matlab Design Point Settings

Design Point	N_h	Training Method
(- -)	1	SCG
(+ -)	3	SCG
(- +)	1	MBR
(+ +)	3	MBR

Cross Validation Responses

The classification 2^2 cross validation results for BPFANN are in Figure 51, and the results for Matlab classification are in Figure 52.

	A	B	C	D	E	F
1			Train		Predict	
2	Nh	W,V	MisClass	3SEM	MisClass	3SEM
3	-	-	3.6%	0.1%	3.6%	0.3%
4	+	-	2.0%	0.06%	2.1%	0.3%
5	-	+	1.6%	0.1%	1.7%	0.2%
6	+	+	4.2%	0.98%	9.4%	1.4%
7		e1	0.5%		3.1%	
8		e2	0.05%		2.7%	
9		e12	1.1%		2.3%	

Fig. 51. Iris BPFANN Classification Cross Validation Results.

BPFANN Cross Validation Commentary

The table in Figure 51 shows a mixed pattern of factor effects and interactions. The obvious best classifier is the smaller network with the uninformative prior showing strong consistency between training and predictive data sets. The design point network (+,+) appears to be clearly over-fitting, while the other design point networks are remarkably consistent though they range somewhat in performance.

	A	B	C	D	E	F
1			Train		Predict	
2	Nh	Mthd	MisClass	3SEM	MisClass	3SEM
3	-	-	2.1%	0.02%	1.6%	0.1%
4	+	-	0.9%	0.01%	4.6%	0.03%
5	-	+	1.9%	0.22%	2.0%	0.91%
6	+	+	1.2%	0.25%	3.6%	1.13%
7		e1	−0.9%		2.3%	
8		e2	0.0%		−0.3%	
9		e12	0.1%		−0.3%	

Fig. 52. Iris Matlab Classification Cross Validation Results.

Matlab Cross Validation Commentary

As with BPFANN, the table in Figure 52 shows the smaller network to be the best classifier in the predictive regime with the desirable pattern that predictive classification is better than it is for training.

Comparison of BPFANN to Matlab

Training performance for Matlab is generally better than BPFANN across design points. Predictive performance is about par for the two different approaches each showing some sensitivity to its design factors (only one of which is common). We note that both systems are well within the traditional figure-of-merit misclassification rate of 5%.

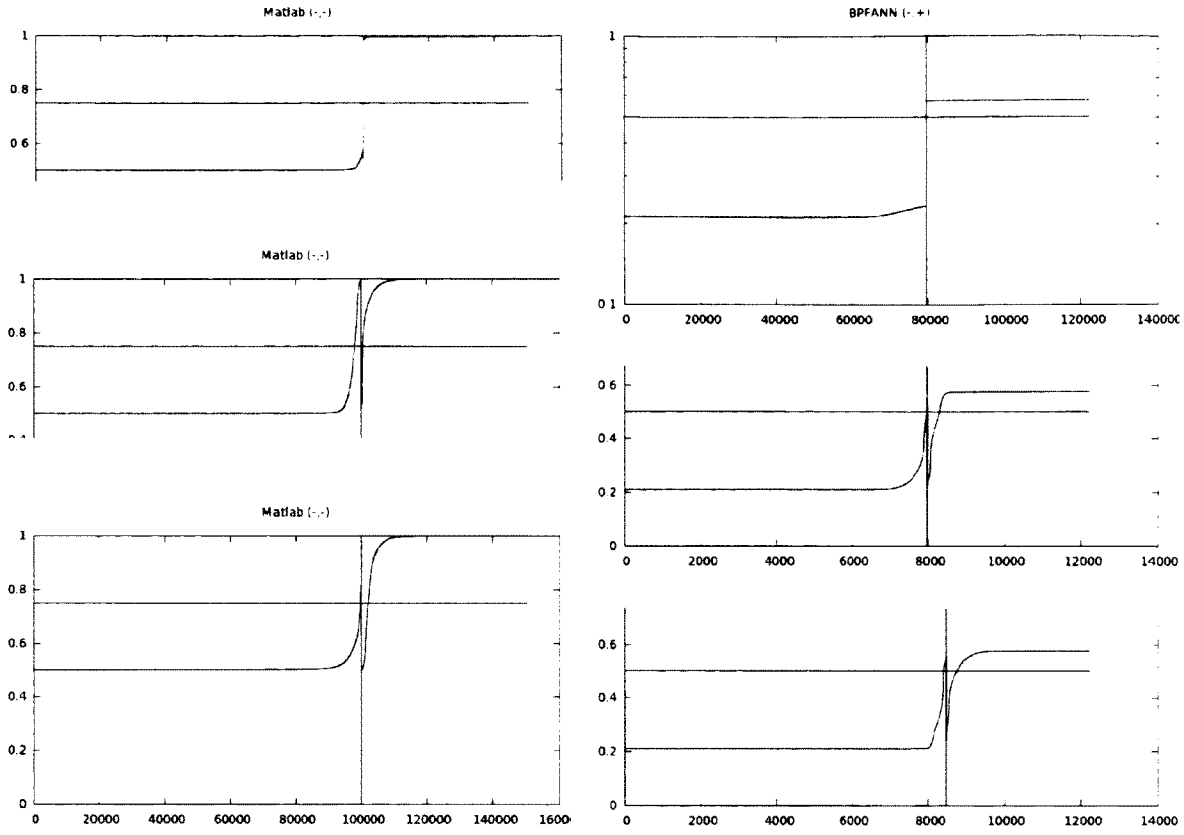


Fig. 53. Iris Misclassification by both Matlab and BPFANN.

In Figure 53 the raw output data for the scoring is for all three classes before softmax output layer transforms are applied (See section 4.2.2). We note that both BPFANN and Matlab are misclassifying a small percentage for the two classes which are not linearly separable.

6.4 CASE STUDY IV CLASSIFICATION

In this case study we analyze a Wisconsin Diagnostic Breast Cancer (WDBC) data set produced by a breast cancer study at the University of Wisconsin Clinical Sciences Center consisting of 569 total patterns with 30 real-valued input features and a single binary classification output which encodes a diagnosis of either malignant or benign. Distribution of diagnoses in these patterns is 357 benign, and 212 malignant.

Input Attributes The input attributes for each pattern are: Three data mean, standard error, and largest value for each of ten real-valued features namely

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness ($perimeter^2/area - 1.0$)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension

of image data for cell nuclei are computed such that for instance, input feature 3 is Mean Radius, input feature 13 is Radius Standard Error, and input feature 23 is Worst Radius.

6.4.1 CROSS VALIDATION

We perform a cross validation study on this data for 1000 trials for both BPFANN (BPD/SRM) and Matlab (SRM) with the parameter setting matrices for both the 2^3 and 2^2 cross validations as listed in Tables 11 and 12 as per sections 5.4.2 and 5.4.1. The results for Matlab classification are in Figure 54, and the results for Matlab classification are in Figure 55.

Table. 11: 2^2 WDBC BPFANN Design Point Settings

Design Point	N_h	$P_0(\boldsymbol{\theta})$
(- -)	1	$\phi(0,1), \phi(0,1), \phi(0,1)$
(+ -)	2	$\phi(0,1), \phi(0,1), \phi(0,1)$
(- +)	1	$\phi(0,100), \phi(0,100), \phi(0,100)$
(+ +)	2	$\phi(0,100), \phi(0,100), \phi(0,100)$

Table. 12: 2^2 WDBC Factorial Matlab Design Point Settings

Design Point	N_h	Training Method
(- -)	1	SCG
(+ -)	2	SCG
(- +)	1	MBR
(+ +)	2	MBR

BPFANN Cross Validation Commentary

The table in Figure 54 shows a clear pattern of increase of misclassification rate for training due to increase in network complexity (effect **e1**) and likewise for prediction. Moving to the broader prior settings decreases the misclassification rate with no appreciable interaction among factors.

	A	B	C	D	E	F
1			Train		Predict	
2	Nh	W,V	MisClass	3SEM	MisClass	3SEM
3	-	-	2.3%	0.1%	2.9%	0.2%
4	+	-	2.8%	0.06%	3.2%	0.2%
5	-	+	2.1%	0.04%	2.4%	0.1%
6	+	+	2.7%	0.05%	2.8%	0.2%
7		e1	0.5%		0.4%	
8		e2	−0.1%		−0.4%	
9		e12	0.0%		0.0%	

Fig. 54. WDBC BPFANN Classification Cross Validation Results.

Matlab Cross Validation Commentary

The table in Figure 54 shows a pattern of decreasing misclassification for training and prediction due to increasing the network complexity (effect **e1**) and increase of misclassification for training and prediction due to use of Bayesian Regulation for training (effect **e2**).

	A	B	C	D	E	F
1			Train		Predict	
2	Nh	Mthd	MisClass	3SEM	MisClass	3SEM
3	-	-	2.2%	0.03%	1.1%	0.1%
4	+	-	1.5%	0.01%	0.1%	0.03%
5	-	+	2.2%	0.03%	1.1%	0.08%
6	+	+	2.4%	0.03%	1.3%	0.01%
7		e1	−0.2%		−0.4%	
8		e2	0.4%		0.6%	
9		e12	0.2%		0.3%	

Fig. 55. WDBC Matlab Classification Cross Validation Results.

Comparison of BPFANN to Matlab

Training performance appears about equal between BPFANN and Matlab; however, Matlab's predictive performance is noticeably superior especially for the more complex network and the Scaled Conjugate Gradient training method. We note that both systems are well within the traditional figure-of-merit misclassification rate of 5%.

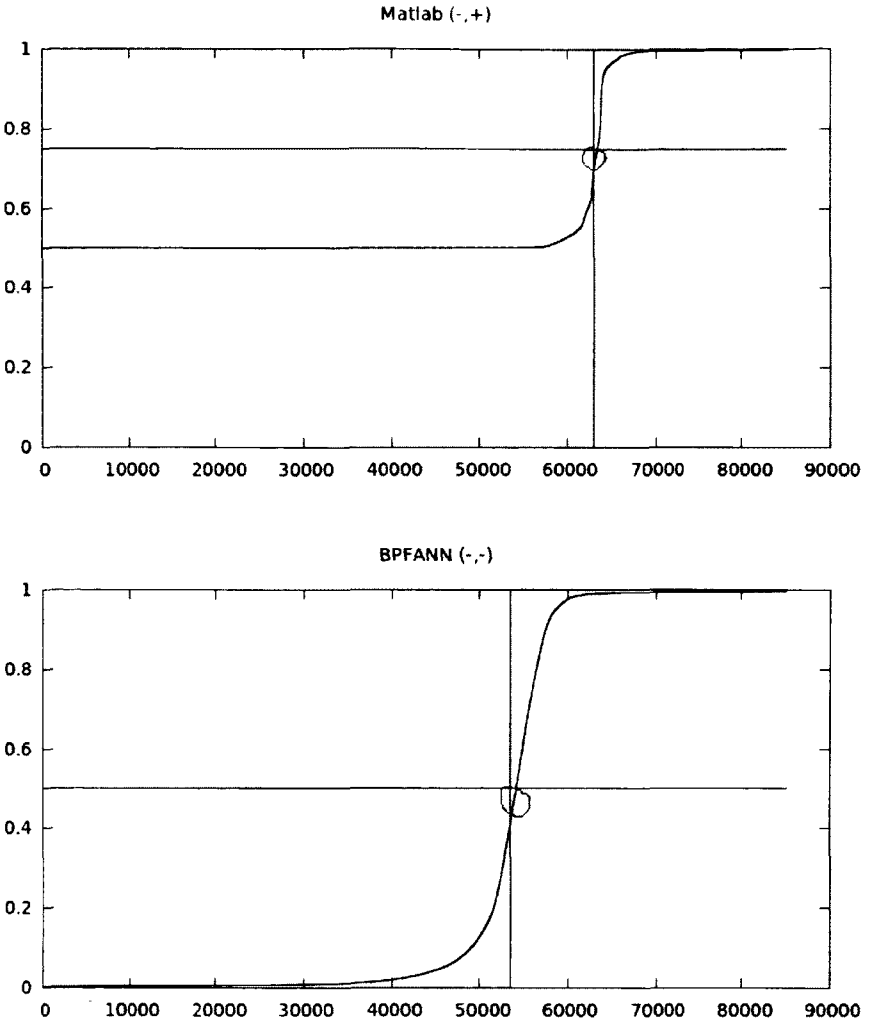


Fig. 56. WDBC Misclassification by both Matlab and BPFANN.

Misclassification regions are the circled area shown for both Matlab and BPFANN in Figure 56. Note that raw data is shown before softmax output layer transforms are applied (See section 4.2.2)

CHAPTER 7

CONCLUSIONS

Cross validation trials show acceptable to good predictive performance for varying levels of precision for all non-linear regression cross validations given that the design points were not pre-optimized to give the best results but were chosen to explore a range of responses for various size networks and other modeling parameters. Performance on simulated data gives additional confidence that performance on real world regression applications will continue to be valuable.

Bayesian predictive distribution modeling via inclusion of the Bayesian Likelihood function parameter as an inferred parameter (i.e., sampled) was successful in regression cross-validation design point factor selections indicating that it is possible to gain accurate predictions at various levels of precision.

Training regularization via the Bayesian prior performs at a high level even for extended data sets as it effectively throttles the MCMC sampling algorithm to parameter scale regions specified by the prior. Prior regularization, however, is no guarantee of good predictive generalization unless the Bayesian prior is selected for parameter sampling regions that demonstrate good modeling performance based on the greatest predictive precision.

SRM predictions also proved accurate to about the same level in the same factor design points that were successful for Bayesian predictive distributions. Given the point estimation nature of SRM and that it is computationally considerably less

expensive than formal Bayesian prediction modeling, this is indicated as a very valuable and cost effective prediction strategy.

Classification performance was also consistently better than the standard figure-of-merit misclassification rate of 5% for all cross validation trials for real world data.

Comparisons to the Matlab ANN Toolkit both regression and classification were favorable given that the Toolkit is production software with many years engineering and refinements by practical use in many diverse fields of application. By contrast, software developed for this research effort is research grade produced by a single individual with limited opportunity for broad application or follow up refinements.

CHAPTER 8

FUTURE RESEARCH

8.1 REDUCED PARAMETER SPACE FORMULATION

For regression problems, requiring the network output (40) to equal the training output data (41) when the network input is (35) we now have

$$QV = T \tag{61}$$

which can be formally solved for the matrix V as

$$V_T = \left[Q^T Q\right]^{-1} Q^T T \tag{62}$$

where we have used the *Moore-Penrose Inverse* for Q and

$$Y = QV_T \tag{63}$$

is an alternate to (40) for the TLP output. Realistically, (61) would be solved with some choice of linear system solver.[1] The primary benefit of (63) is that the weights in the matrix V have been entirely eliminated from any sampling considerations thus reducing the dimension of the parameter space to be sampled by $(N_h + 1)N_o$. We

[1]This formulation is obviously sensitive to the ability of the chosen matrix solution methodology (e.g., Singular Value Decomposition) to average out the effects of the stochastic portion of the given data T.

now have the choice of TLP output from (40) or (63) depending on whether we wish to sample for the components of V or not. We stress that while the particular matrix simulation articulated above is for a TLP, it can be replaced for ANNs with a non-TLP structure; the rest of the methodology is not in any way dependent on choice of (feed-forward, supervised) ANN structure. In such cases it may not in general be possible to reduce the parameter space to be sampled.

8.2 GENERALIZED NORMAL BASED LIKELIHOOD FUNCTION

The SRM could model the stochastic residue of the training regimen using the Generalized Normal Density

$$f(x \mid \mu, \alpha, \beta) = \frac{\beta}{2\alpha\Gamma\left(\frac{1}{\beta}\right)} exp\left(-\frac{|x-\mu|}{\alpha}\right)^{\beta} \; ; \; \alpha, \beta > 0 \tag{64}$$

with mean μ and variance $\frac{\alpha^2\Gamma\left(\frac{3}{\beta}\right)}{\Gamma\left(\frac{1}{\beta}\right)}$, or in order to possibly model any residual skew and kurtosis in the training/predictive residual.

8.3 MCMC METROPOLIS HASTINGS SAMPLING

The implications of using the Metropolis Hastings Markov Chain Monte Carlo (MHMCMC) sampling algorithm for the sampling of ANN parameter vectors (UoD) could be studied. MHMCMC requires use of a "correcting" distribution (to gain efficiency), and we propose that the prior distribution, however determined, is the only logical candidate as a matter of principle.

8.4 CROSS ENTROPY EVALUATION

Here we explore the prospect of an alternate Cross Entropy based solution that makes no reference to Bayesian probability.

8.4.1 SHANNON'S INFORMATION THEORY

Several years after Cox's paper[9], in 1948 Claude Shannon published his famous paper [48] on communication theory. In this paper Shannon was able to reason out in a manner not too dissimilar from Cox's, a measure of probabilistic uncertainty[2]. In so doing, he recovered the entropy formula of the Statistical Mechanics (SM) of J. W. Gibbs, namely that $H = -k \sum_p p \cdot log(p)$. In words, the weighted value of $log(p)$ over all accessible states.

8.4.2 JAYNES PRINCIPLE OF MAXIMUM ENTROPY

Physicist E.T. Jaynes [25] took note of both Shannon's entropy result and Cox's Axioms and subsequently caused a stir in the physics community by reformulating SM as a problem of inference under incomplete information of a physical system arguing for the information theoretic nature of such inference and also of entropy. Jaynes took the view that probability distributions were encoders of information concerning a logical state of incomplete knowledge with uncertainty in the Coxian sense. Jaynes in the same publication also established the Principle of Maximum Entropy (PME) as a conditioning requirement for probability distributions in order

[2]The first use of the entropy measure for uncertainty was apparently due to Hartley [21] who used $log_2 N$ for N equally uncertain outcomes. This is a special case for N equally likely outcomes of the Gibbs-Shannon formula.

to achieve that proper state of knowledge of maximum but constrained uncertainty or, in his words, "to be maximally non-committal to what is not actually known," and which was to serve as an essential inferential heuristic of researcher honesty and ethical rationality. This theorem[3] is vital in cases where there is a second level of uncertainty where not only is the point solution underdetermined but the uncertainty over a nexus of point solutions - expressed as a probability distribution - is itself not unique (underdetermined). In such cases, Jaynes councils us that the honest thing to do, for all such distributions, is choose the one with the largest entropy so that we discriminate among alternative models as little as the evidence (data) permits under the circumstances.

Principle of Maximum Entropy Consider a parameterized system U that has some discrete parameter state space S available to it under the conditions that certain *macroscopic* values a_c, $c = 1, .., N_c$ are known. By *macroscopic* we mean that there exist N_c scalar valued functions $f_c(s)$ of the discrete state space index $s \in S$, where the value for the function $f_c(s)$ for the state s is designated as f_{cs} and each macroscopic value a_c is the *state space probabilistic expectation* of the related function $f_c(S)$. With p_s as the *probability*[4] that the system is in the state s we have

$$\mathbf{p}F = \mathbf{a} \Rightarrow \sum_s [p_s f_{cs}] = \langle f_c(S) \rangle \equiv a_c; \; c = 1, .., N_c. \tag{65}$$

[3]Alternatively, it is an heuristic.
[4]uncertainty

Now consider a probability mass function $P = \{s, p_s\}$ for the state space S commensurate with (65). In general[5] $\{s, p_s\}$ is *underdetermined* by equation (65) and so the following question naturally arises: *what is the proper selection of* $\{p_s\}$ *for the many possible* $\{s, p_s\}$ *consistent with equation (65) over the state space* S*?* The key to the solution is in the characterization of the term *proper*. If we understand proper as *least committal to what is not actually known*, then we can obtain $\{s, p_s\}$ by maximizing the *informational entropy* of $\{s, p_s\}$ defined[48] as:

$$H(P) \equiv - < log(p) >= - \sum_s p_s log(p_s) = -\mathbf{p} \cdot log\,(\mathbf{p}) \tag{66}$$

subject to (65). Thus we have a *constrained optimization problem.* It was just this problem that was solved analytically by physicist J. Willard Gibbs yielding the *Gibbs Distribution* unique for the stated conditions[6]:

$$\mathbf{p} = Z^{-1} exp\,(\boldsymbol{\lambda} F) \Rightarrow p_s = Z^{-1} exp\left[\sum_{c=1}^{N_c} \lambda_c f_{cs}\right] \tag{67}$$

where the normalizing constant Z is[7] given as

$$\sum_s p_s = 1 \rightarrow Z = \sum_s exp\left[\sum_{c=1}^{N_c} \lambda_c f_{cs}\right] \tag{68}$$

[5]Whenever $N_C <$ size of the state space S.

[6]Gibbs apparently saw his effort as characterizing the population of different micro-states for given macroscopic properties of a system in thermodynamic equilibrium and in a state of maximum (physical) entropy. E.T. Jaynes realized that Gibbs had in fact solved a general problem of inference under uncertainty and that entropy is information theoretic in nature such that the inference necessarily is not a function of any information not given, unlike classical or orthodox statistics in particular which conditions its inferences on data not given or observed.

[7]Known in Statistical Mechanics as the *Partition Function*

and the vector $\boldsymbol{\lambda}$ is given formally by

$$\frac{\partial}{\partial \lambda_c} log\left(Z\left(\boldsymbol{\lambda}\right)\right) = a_c. \tag{69}$$

Principle of Minimum Cross Entropy Consider a normalized discrete probability mass function $Q = \{\{s_1, q_1\}, .., \{s_{N_s}, q_{N_s}\}\}$ for some N_s states of a system U indexed by s such that

$$\sum_s q_s = 1 \Longrightarrow \sum_s q_s - 1 = 0. \tag{70}$$

Sans any other information, according to Laplace's *Principle of Insufficient Reason*, we must make the assignment[8]

$$q_s = \frac{1}{N_s} \tag{71}$$

consistent with the *normalization constraint*, which though a primitive constraint, is all that is necessary to determine a unique probability distribution. Suppose we also have additional constraint information of the general form:

$$\sum_s f_{cs} q_s = a_c;\ c = 1, .., N_c \Longrightarrow F\mathbf{q} - \mathbf{a} = 0. \tag{72}$$

[8]Note that this can also be found by application of equation (67) where normalization is the only available constraint.

Instead of the *entropy* of Q

$$H(Q) \equiv -\sum_s q_s log(q_s) = -\mathbf{q} \cdot log\,(\mathbf{q}) \tag{73}$$

we consider the *cross entropy*[2] of Q with some given *prior* P over a common state space S:

$$H_{Q|P}(Q|P) \equiv E_Q \left\{ log\left(\frac{q}{p}\right) \right\} = \sum_s q_s log\left(\frac{q_s}{p_s}\right). \tag{74}$$

In seeking the *minimum* of equation (74), we are finding the distribution Q that is as close to P as possible and is commensurate with (70) and (72). This is to be understood as a *conservative learning law* in the sense that it seeks to obtain the *minimum possible* change to P to obtain Q. Mathematically this is accomplished by determining Q as minimizing (74) subject to (72). By inspection of (71) and (74) we can see that the Principle of Maximum Entropy is a special case of the Principle of Minimum Cross Entropy when the reference (starting) distribution in the PME is the *uniform* distribution. We use the method of Lagrange Multipliers to minimize (74) subject to (72), and (70) with Lagrangian

$$L(Q, \boldsymbol{\lambda}, \lambda_0) = H(Q|P) + \boldsymbol{\lambda} \cdot (F\mathbf{q} - \mathbf{a}) + \lambda_0 \left(\sum_s q_s - 1 \right) \tag{75}$$

so that

$$\nabla_{Q,\boldsymbol{\lambda},\lambda_0} L(Q, \boldsymbol{\lambda}, \lambda_0) = 0 \tag{76}$$

[2]Cross Entropy is inconsistently defined across the literature both in algebraic form and sign.

yielding

$$\frac{\partial}{\partial q_s}\sum_k q_k log\left(\frac{q_k}{p_k}\right) = \lambda_0\frac{\partial}{\partial q_s}\sum_k q_k + \sum_c^{N_c}\lambda_c\frac{\partial}{\partial q_s}\sum_k f_{ck}q_k \tag{77}$$

for say the equation concerning the state s for Q implied by (76), where we must determine $\{\boldsymbol{\lambda}, \lambda_0\}$ to obtain a solution. Equation (77) reduces to

$$log\left(\frac{q_s}{p_s}\right) + 1 = \lambda_0 + \sum_c^{N_c}\lambda_c f_{cs}. \tag{78}$$

Thus,

$$q_s = p_s exp\left(\lambda_0 - 1\right) exp\left(\sum_c^{N_c}\lambda_c f_{cs}\right). \tag{79}$$

As λ_0 is clearly related to normalization, we absorb it into a normalization constant and write:

$$q_s \propto p_s exp\left(\sum_c^{N_c}\lambda_c f_{cs}\right) \tag{80}$$

All that remains is to determine $\{\boldsymbol{\lambda}, \lambda_0\}$ via agreement with equations (72) and (70) using (79).

8.4.3 CROSS ENTROPY EVALUATION

In all cases of use of MOEs and MOPs, we believe it to be feasible for some networks to explore use of the PME to independently transform the prior to posterior directly as an independent comparison. We believe it is possible to develop a Minimum Cross Entropy' parameter distribution for ANNs based on the powerful ability of the maximum entropy distribution to discriminate[27] among competing models

yet at the same time in the least possible discriminatory fashion possible[27], leaving the most possible shaping influence for new data. The basic procedure would be to apply (70), (72) and (80) to the case of an arbitrary ANN. The way to do this is to write the constraint values f_{cs} in (72) in terms of the network output data Y, and the constraint equation constants (given) using the training[9] (output[10]) data matrix T which has the same dimensions as the network simulation output matrix Y. This implies that the f_{cs} in equation (72) are in fact the y_{pk} of equation(3) such that the constraint index c ranges over the dimensions of the output Y. That is, c ranges over both p and k of equation(3). Therefore, writing the f_{cs} in equation(3) now as f_{pks} we have

$$f_{pks} \equiv y_{pks} = ann(p,k,s) = v_{0ks} + \sum_{h=1}^{N_h} v_{hks}\psi\left(w_{0hs} + \sum_{i=1}^{N_i} w_{ihs}x_{pi}\right), \qquad (81)$$

and the constraint constants a_c in (72) now written as a_{pk} are now the corresponding elements of the training data output matrix t_{pk}. Thus, we require

$$\sum_s [q_s y_{pks}] \cong t_{pk} \; ; \; n = 1,..,N_p, k = 1,..,N_o \qquad (82)$$

as the specific form of (72). Note that the network parameter matrices W, V now are indexed by state since the state space we are sampling over requires distinct values for the network parameters (yielding distinct components for the output matrix Y) which

[9]We are exploring a Maximum Entropy *inference* here. In our case the given data are our *observations*

[10]We are of course using the training data input matrix as the desired or *target* network simulation output matrix Y throughout.

are encapsulated in the combined parameter matrices W, V. It is important to note that equation (82) almost gets us there but not quite. The rub is that the training data T is in general contaminated with noise, so we will have to deal with that contingency. One way to deal with it is to realize that the expected network output which is the LHS of equation (82), will, depending upon choice of the complexity of the network, reflect some *model based generalization*, so it will not in general be the same as T. The magnitude of $T - E(Y)$ will, however, be our estimate of the residual noise power in the data, so the basic strategy is to find a solution for $\boldsymbol{\lambda}$ that minimizes it.

Numerical Method Solution A matrix solution involves recognizing equation (82) as an instance of the ubiquitous $A\mathbf{x} = \mathbf{b}$ linear system and use one of the many numerical methods for it. The essential caveat is that we would need to solve the system using the noise contaminated T at least at first as our best and perhaps only estimate for the true model.

Simulation Based Solution An alternate, albeit more brute force solution, would be to use *Monte Carlo simulation* to sample the components of $\boldsymbol{\lambda}$ and iterate until the magnitude of the residual noise vector $T - E(Y)$ reaches a steady state minimum, non-terminating simulation style.

8.5 MINIMUM CROSS ENTROPY MOMENT PRIOR

We believe it should be possible to develop a minimum cross entropy moment based prior distribution for ANNs. The basic procedure would be to replace the constraints in equation (72) with certain other macroscopic properties in the form of distribution moments of the training data descriptive statistics. We could use, for example, the first several moments of the training output, e.g. mean, variance, etc. In this case, for a one dimensional output vector we rewrite equation (72) with just two constraints: the first is the mean of the simulated network output for each dimension of the output; the second is the variance; e.g., for the mean we write

$$E(\mathbf{y}_k) = \sum_s \left[q_s \sum_n^{N_p} y_{nks} \right] \simeq \sum_n^{N_p} t_{nk} = E(\mathbf{t}_k) \; ; \; k = 1, .., N_o \tag{83}$$

and similarly for the variance

$$E(E(\mathbf{y}_k)^2) - (E(\mathbf{y}_k))^2 \simeq E(E(\mathbf{t}_k)^2) - (E(\mathbf{t}_k))^2 \; ; \; k = 1, .., N_o \tag{84}$$

with $\mathbf{q}$ to satisfy both the moment constraints for the output vector mean (83) and output vector variance (84). For an output matrix of column dimension N_o, we have N_o such pair equations. Even more compelling is to first normalize[11] the data such that a minimum cross entropy prior could be produced that could serve as an Objective Bayesian reference prior distribution good for all ANN modeling efforts with the same input training data but arbitrary output training data.

[11]Offset and rescale the training data so that its mean and standard deviation of each output dimension over all training patterns are 0,1 respectively. This is a standard technique in ANN error backpropagation training.

BIBLIOGRAPHY

[1] A. Law and W. Kelton, *Simulation Modeling and Analysis*, New York, McGraw-Hill. 2003.

[2] S. E. Ahmed and N Reid, *Empirical Bayes and Likelihood Inference* New York, Springer, 2001.

[3] J. Banks, J. S. Carson II, B. L. Nelson and D. M. Nicol, *Discrete Event System Simulation*, New York, Prentice Hall, 2001.

[4] J. M. Bernardo and A. F. M. Smith, *Bayesian Theory*, New York Wiley, 1994.

[5] S. Brooks and G. O. Roberts, "Assessing Convergence of Markov Chain Monte Carlo Algorithms", *Statistics and Computing*, vol. 8, pp. 319-335, 1997.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning*, New York, Springer, 2006.

[7] W. Buntine and A. Weigend, "Bayesian Backpropagation", *Complex Systems* vol. 5, pp. 603-643. 1991.

[8] B. M. Ozyildirim and M. Avci, "Generalized Classifier Neural Network", *Neural Networks*, vol. 39, pp. 1826, 2013.

[9] R. T. Cox, "Probability, Frequency, and Reasonable Expectation", *Am. Jour. Phys.*, vol. 14, pp. 113, 1946.

[10] R. T. Cox, *The Algebra of Probable Inference*, Baltimore, The Johns Hopkins Press, 1961.

[11] G. Cybenko, "Approximation by Superpositions of Sigmoidal Functions". *Math Control Signals Sys.* vol 2, pp. 303-314, 1989.

[12] A. P. Dawid, M. Stone and J. V. Zidek, "Marginalization Paradoxes in Bayesian and Structural Inference", *Jour. Royal Stat. Soc. B*, vol. 35, no. 2, pp. 189-233, 1973.

[13] S. Dreiseitla and L. Ohno-Machado, "Logistic Regression and Artificial Neural Network Classification Models: A Methodology Review", *J. Bio Info*, vol. 35, pp. 352359, 2002.

[14] P. J. Edwards, A. F. Murray, G. Papdopoulos, A. R. Wallace, J. Barnard, and G. Smith, "The Application of Neural Networks to the Papermaking Industry", *IEEE Trans. Neural Networks* vol 10, pp. 1456-1464, 1999.

[15] B. Farley and W. A. Clark, "Simulation of self-Organizing Systems by Digital Computer", *IEEE Trans. Info. Theory*, vol. 4, pp.76-84, 1954.

[16] B. G. Farley, "Self-Organizing Models for Learned Perception" in M.C. Yovits and S. Cameron (editors), *Self-Organizing Systems*, Oxford, Pergamon Press, 1960.

[17] R. Fischer, "Moments and Product Moments of Sampling Distributions for R". *London Math. Soc.*, vol 2(3), pg. 199, 1929.

[18] D. Foresee and M. Hagan, Gauss-Newton Approximation to Bayesian Learning. *Proc. of the 1997 Int. Joint Conf. on Neural Networks*, vol. 3, pp. 1930 - 1935, 1997.

[19] R. Gencay and M. Qi, "Pricing and Hedging Derivative Securities with Neural Networks; Bayesian Regularization, Early Stopping, and Bagging", *IEEE Trans. Neural Networks*, vol. 12, pp. 726-734, 2001.

[20] M. S. Goodrich, "Markov Chain Monte Carlo Bayesian Learning for Neural Networks", ModSimWorld, Hampton VA, Conference 2010, pp. 268-277.

[21] R. V. Hartley, "Transmission of Information", *Bell Systems Technical Journal*, vol 7, pp. 535-563, 1928.

[22] D. O. Hebb, *The Organization of Behavior*, John Wiley & Sons, New York, 1949.

[23] K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feed-forward Networks Are Universal Approximators", *Neural Networks*, vol. 2, pp. 359-366, 1989.

[24] D. Husimer, W. D. Penny, and S. J. Roberts, "An Empirical Evaluation of Bayesian Sampling with Hybrid Monte Carlo for Training Neural Network Classifiers". *Neural Networks*, vol. 12, pp. 677-705, 1999.

[25] E. T. Jaynes, "Information Theory and Statistical Mechanics", *The Physical Review*, vol. 106, no. 4, pp. 620-630, 1957.

[26] E. T. Jaynes, *Probability Theory The Logic of Science*, Cambridge University Press, 2003.

[27] E. T. Jaynes, "On the Rationale of Maximum Entropy Methods", *Proc. of the IEEE*, vol. 70, no. 9, Sept 1982.

[28] R. E. Kass and L. Wasserman, "The Selection of Prior Distributions by Formal Rules", *J. Amer. Stat. Assoc.* vol. 91, no. 435, pp. 1343-1370, 1996.

[29] Y. LeCun, "Learning Processes in an Asymmetric Threshold Network", in E. Bienenstock, F. Fogelman-Souli, and G. Weisbuch, G. (Eds), *Disordered Systems and Biological Organization*, pp. 233-240, Springer-Verlag, Les Houches, France, 1986.

[30] Y. LeCun, L. Bottou, G. B. Orr, and K. Muller, "Efficient Backpropagation", *Neural Networks: Tricks of the Trade*, Springer, 1998.

[31] H. K. H. Lee, "A Noninformative Prior for Neural Networks", *Machine Learning*, vol. 50, pp. 197-212, 2003.

[32] D. J. C. MacKay, "Bayesian Interpolation", *Neural Computation*, vol. 4, pp. 415-447, 1992.

[33] D. J. C. MacKay, "A Practical Bayesian Framework for Backpropagation Networks", *Neural Computation*, vol. 4, pp. 448-472, 1991.

[34] D. J. C. MacKay, "The Evidence Framework Applied to Classification Networks". *Neural Computation*, vol 4, pp. 720-736, 1992.

[35] D. J. C. MacKay, "Probable Networks and Plausible Predictions-a Review of Practical Bayesian Methods for Supervised Neural Networks", *Neural Computation*, vol. 11, pp. 1035-1068, 1995.

[36] R. L. Mattson, "The Design and Analysis of an Adaptive System for Statistical Classification", S.M. Thesis, MIT May 22, 1959.

[37] M. C. Medeiros, A. Veiga, A. and C. E. Pedreira, "Modeling Exchange Rates: Smooth Transitions, Neural Networks and Linear Models". *Neural Networks*, vol. 12, pp. 755-764, 2001.

[38] M. L Minsky and S. Papert, *Perceptrons, An Introduction to Computational Geometry*, Boston, MIT Press, 1969.

[39] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.

[40] W. S. McCulloch and W.H. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-137, 1943.

[41] R. M. Neal, *Bayesian Learning for Neural Networks*, New York, Springer, 1996.

[42] D. B. Parker, "A Comparison of Algorithms for Neuron-Like Cells", in J.S. Denker (Ed.), *Neural Networks for Computing*, pp 327-332, New York, American Institute of Physics, 1986.

[43] N. Rochester, J. H. Holland, L. H. Haibt, and W. L. Duda, "Tests on a Cell Assembly Theory of the Action of the Brain Using a Large Digital Computer", *IEEE Trans. of Info. Theory*, vol. 2, pp. 80-93, 1956.

[44] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", *Psychological Review*, vol. 65, pp. 386-408, 1958.

[45] F. Rosenblatt, "Perceptron Simulation Experiments", *Proc. IEEE*, vol. 48, pp. 301-309, 1960.

[46] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan books, Washington D.C., 1962.

[47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors". *Nature*, vol. 323, pp. 533-536, 1986.

[48] C. E. Shannon, "A Mathematical Theory of Communication", *Bell System Technical Journal*, vol. 27, pp. 379-423, 1948.

[49] J. E. Shore and R. W. Johnston, "Axiomatic Derivation of the Principle of Maximum Entropy and the Principle of Minimum Cross Entropy", *IEEE Trans. Info. Theory*, vol. 26, pp. 26-37, 1980.

[50] M. Tribus, *Rational Descriptions, Decisions and Designs*, Pergamon Press, 1969.

[51] M. E. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine", *J. Machine Learning*, vol. 1, pp. 211-244, 2001.

[52] K. S. Van Horn, "Constructing a Logic of Plausible Inference: a Guide to Coxs Theorem", *International Journal of Approximate Reasoning*, vol. 34, Issue 1, September, pp. 324, 2003.

[53] F. Vivarelli and C. K. I. Williams, "Comparing Bayesian Neural Network Algorithms for Classifying Segmented Outdoor Images", *Neural Networks*, vol. 11, pp. 427-437, 2001.

[54] P. J. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences", Ph.D. dissertation, Harvard University, 1974.

[55] B. Widrow, "Generalization and Information Storage in Networks of Adaline Neurons". In M.C. Yovits, G.T. Jacobi, and G.D. Goldstein (Eds.), *Self-Organizing Systems*. Washington D.C., Spartan Books. 1962.

[56] D. H. Wolpert, "On the Use of Evidence in Neural Networks", *Advances in Neural Information Processing Systems 5*, Kauffman Publishers, 1993.

[57] C. P. I. van Hinsbergen, J. W. C. van Lint, and H. J van Zuylen, "Bayesian Committee of Neural Networks to Predict Travel Times with Confidence Intervals", *Transportation Research Part C*, vol. 17, pp. 498-509. 2009.

[58] I. C. Yeh, "Modeling Slump Flow of Concrete Using Second-order Regressions and Artificial Neural Networks", *Cement and Concrete Composites*, vol. 29, no. 6, pp. 474-480, 2007.

[59] V. Agrawal and A. Sharma, "Prediction of Slump in Concrete using Artificial Neural Networks, *World Academy of Science, Engineering and Technology*, vol. 4, 2010.

VITA

Michael. S. Goodrich
Department of Modeling and Simulation
Old Dominion University
Norfolk, VA 23529

EDUCATION

Doctor of Philosophy, Engineering with a concentration in Modeling and Simulation, Old Dominion University, Norfolk, VA, May 2014

Master of Science, with a concentration in Physics, Old Dominion University, Norfolk, VA, May 1992

Bachelor of Science, with a concentration in Physics, Old Dominion University, Norfolk, VA, May 1979

SELECTED PUBLICATIONS

Goodrich, M.S., *Markov Chain Monte Carlo Bayesian Learning for Neural Networks.* ModSimWorld 2010.

Garcia, C. and Goodrich, M.S., *Effects of Health Care Policy Decisions on Physician Availability.* ModSimWorld 2010.

CONFERENCE PRESENTATIONS

Goodrich, M.S., *Markov Chain Monte Carlo Bayesian Learning for Neural Networks.* ModSimWorld 2010.

Garcia, C. and Goodrich, M.S., *Effects of Health Care Policy Decisions on Physician Availability.* ModSimWorld 2010.

Typeset using LaTeX.