UNIVERSITY OF CALIFORNIA SAN DIEGO

Strategic Monte Carlo and Variational Methods in Statistical Data Assimilation for Nonlinear Dynamical Systems

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Physics

by

Aleksandra Shirman

Committee in charge:

Professor Henry Abarbanel, Chair
Professor Patrick Diamond
Professor Olga Dudko
Professor Jeffrey Elman
Professor Michael Holst

2018

The dissertation of Aleksandra Shirman is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California San Diego

2018

DEDICATION

To my parents who inspired me to be a scientist.

And to my fiance, Blake, who supported me through all the ups and downs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I want to thank Henry Abarbanel, my advisor, for giving me so much feedback and helping me to develop the tools I need to be successful. Henry's endless curiosity and drive has been an inspiration to me. I am grateful for his confidence in my ability and future as a scientist and the push he constantly gives me to search for my own answers.

I want to thank Eve Armstrong for being such a great role model and always answering my questions and giving me advise on every topic. Her questions and comments challenged me and helped to guide and organize my thoughts and understanding. Thanks to all the other graduate students in the Abarbanel research group for their questions, comments, critiques and the stimulating conversations.

I would also like to thank the ARCS foundation for their support.

Finally thank you to my parents who have always supported me and inspired me, thank you to my friends who helped me relax and unwind and thank you to my fiance, Blake, for being there to keep me sane throughout graduate school. I don't know how I would have done it without your love and support.

Chapter 2, in part, is being prepared for publication of the material as it may appear in S. Shirman, H. D. I. Abarbanel, Strategic Monte Carlo Methods for State and Parameter Estimation in High Dimensional Nonlinear Problems. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, has been adapted from the material submitted for publication as it may appear in H. D. I. Abarbanel, P. J. Rozdeba, and S. Shirman, Machine Learning; Deepest Learning as Statistical Data Assimilation Problems, Neural Computation. The dissertation author was a coauthor on this paper.

Chapter 4, in part, contains work that will be prepared for publication in the future in H.D.I. Abarbanel, G. Silva,S. Shirman, V. George, A. Gupta, Z. Fang, P. Rozdeba, A. Ty and R. Gonzalez. The dissertation author was a collaborator on this work.

VITA

2013	B. A. in Physics, University of California Berkeley
20013-2014, 2018	Graduate Teaching Assistant, University of California San Diego
2013-2018	Graduate Student Researcher, University of California San Diego
2016	M. S. in Physics, University of California San Diego
2018	Ph. D. in Physics, University of California San Diego

PUBLICATIONS

S. Shirman, H.D.I. Abarbanel, Strategic Monte Carlo Methods for State and Parameter Estimation in High Dimensional Nonlinear Problems, *arXiv preprint arXiv:1805.09838. (in preparation)*.

H.D.I. Abarbanel, P. Rozdeba, & S. Shirman (2017). Machine Learning, Deepest Learning: Statistical Data Assimilation Problems, *arXiv preprint arXiv:1707.01415. (Neural Computation; in review)*

H.D.I. Abarbanel, S. Shirman., D. Breen, N. Kadakia, D. Rey, E. Armstrong, and D. Margoliash. (2017). A unifying view of synchronization for data assimilation in complex nonlinear networks *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12), 126802.

ABSTRACT OF THE DISSERTATION

Strategic Monte Carlo and Variational Methods in Statistical Data Assimilation for Nonlinear Dynamical Systems

by

Aleksandra Shirman

Doctor of Philosophy in Physics

University of California San Diego, 2018

Professor Henry Abarbanel, Chair

Data Assimilation (DA) is a method through which information is extracted from measured quantities and with the help of a mathematical model is transferred through a probability distribution to unknown or unmeasured states and parameters characterizing the system of study. With an estimate of the model paramters, quantitative predictions may be made and compared to subsequent data.

Many recent DA efforts rely on an probability distribution optimization that locates the most probable state and parameter values given a set of data. The procedure developed and demonstrated here extends the optimization by appending a biased random walk around the states

and parameters of high probability to generate an estimate of the structure in state space of the probability density function (PDF). The estimate of the structure of the PDF will facilitate more accurate estimates of expectation values of means, standard deviations and higher moments of states and parameters that characterize the behavior of the system of study. The ability to calculate these expectation values will allow for an error bar or tolerance interval to be attached to each estimated state or parameter, in turn giving significance to any results generated. The estimation method's merits will be demonstrated on a simulated well known chaotic system, the Lorenz 96 system, and on a toy model of a neuron. In both situations the model system provides unique challenges for estimation: In chaotic systems any small error in estimation generates extremely large prediction errors while in neurons only one of the (at minimum) four dynamical variables can be measured leading to a small amount of data with which to work.

This thesis will conclude with an exploration of the equivalence of machine learning and the formulation of statistical DA. The application of previous DA methods are demonstrated on the classic machine learning problem: the characterization of handwritten images from the MNIST data set. The results of this work are used to validate common assumptions in machine learning work such as the dependence of the quality of results on the amount of data presented and the size of the network used. Finally DA is proposed as a method through which to discern an 'ideal' network size for a set of given data which optimizes predictive capabilities while minimizing computational costs.

Chapter 1

Introduction

Scientific research in many fields from geoscience to neuroscience involves making estimates of physical parameters and the state of a system using knowledge of only a subset of the system state variables. The use of only a subset of the state variables within the system for parameter estimation is because it is rare that every degree of freedom that is necessary to describe a system can be recorded simultaneously. In such a situation when only a subset of the degrees of freedom are known, the full state of the system must be inferred from the measured subset. This inference consists of a transfer of information from measured to unmeasured degrees of freedom through constraints determined by a mathematical model of the system of study leading to the generation of estimates for unknown states and parameter values. These estimates allow for the complete characterization of the system of study and for the generation of numerical predictions to validate the results. Methods of information transfer between measured and unmeasured degrees of freedom make up the field of Data Assimilation (DA) [22].

This thesis discusses the methods through which the transfer of information may be accomplished as well as demonstrating some real world examples of this transfer of information. We apply the methods of information transfer, in particular, to systems that may be described by a set of couple differential or difference equations, specifically those in which the dependence

on the time is implicit. The application of these methods is done with the goal of estimating unknown quantities within the system in such a way as to be able to quantitatively predict future time evolution of the system.

1.1 The Problem to be Solved

Consider a system in which the degrees of freedom are defined as the components of a D-dimensional vector, $\mathbf{x}$. The evolution of $\mathbf{x}$ can be described by the equations

$$\dot{\mathbf{x}}_i(s) = \mathbf{f}_i(\mathbf{x}(s), \mathbf{p}) \tag{1.1}$$

$$\mathbf{x}_i(s+\Delta s) = \mathbf{F}_i(\mathbf{x}(s), \mathbf{p}) \tag{1.2}$$

where s is the independent variable and $\mathbf{p}$ is an N_p-dimensional vector of all constant parameters of motion. The function, $\mathbf{f}(\cdot)$, in Eq. (1.1) defines the equations of motion of the states, $\mathbf{x}$. The function, $\mathbf{F}(\cdot)$, in Eq. (1.2) is the time discretized version of $\mathbf{f}(\cdot)$. For notational simplicity we drop the explicit dependence on the parameters, $\mathbf{p}$, in the expressions for the functions, $\mathbf{f}(\cdot)$ and $\mathbf{F}(\cdot)$. The discretization is defined as

$$\mathbf{F}(\mathbf{x}(s)) = \mathbf{x}(s) + \int_s^{s+\Delta s} \mathbf{f}(\mathbf{x}(w))dw \tag{1.3}$$

DA methods for information transfer are typically applied to problems which can not be analytically solved and thus the integral in Eq. (1.3) is difficult or impossible. As such, the integral for calculating $\mathbf{F}(\cdot)$ is approximated through an arbitrary choice of discretization method.

Let us now consider a system in which only an L-dimensional subset of the D-dimensional state variable vector can be measured. We define the vector of measurement values at time s as $\mathbf{y}(s)$. Acknowledging that measurements are not noiseless and that in some situations the

measurement may be of a function of multiple state variables, we define $\mathbf{y}(s)$ in the following manner:

$$\mathbf{y}(s) = \mathbf{h}(\mathbf{x}(s)) + \boldsymbol{\eta}_m(s) \tag{1.4}$$

The function, $\mathbf{h}(\cdot)$ defines the measurement function. If the measured values are a subset of state variables, then $\mathbf{h}(\cdot)$ is simply a projection from D-dimensional to L-dimensional space. The vector, $\boldsymbol{\eta}_m(s)$, is a vector of measurement noise. Assuming uncorrelated Gaussian measurement error, we define $\boldsymbol{\eta}_m(s)$ as a vector whose elements are sampled from Gaussian distributions centered at zero and with standard deviation determined by the accuracy of the measurement for the corresponding degree of freedom. The value of the standard deviation of each Gaussian is due to the experimental accuracy and without replacing experimental equipment can not be reduced.

If one wishes to characterize the state of the system and predict future behavior, it is necessary to generate an estimate of all D state variables, including the $D-L$ state variables that have not been measured, as well as the N_p parameters that govern the equations of motion of the full state of the system. In a sense, to estimate the values of $\mathbf{x}(s)$ and $\mathbf{p}$ in such a way as to best fit Eq. (1.3) to the measured experimental data. For complex and high dimensional systems with many unknown states and parameters this is a difficult problem. This thesis will address computational methods for making such estimations and will present examples of their application.

To do this, we define a probabilistic framework in which to approach the problem. We wish to determine an expression for the probability of the full state of the system at the measured time points and the parameters of motion conditioned on the measured data and the structure of the model. We can then use this probability to determine the expectation value of any quantity we wish to estimate.

To simplify notation, we define the path or set of state variables at all time points within the estimation window as $\mathbf{X}(s_f) = \{\mathbf{x}(0), \mathbf{x}(s_1), \ldots, \mathbf{x}(s_f)\}$ and the full path of data or the set of

measurements at all time points within the estimation window as $\mathbf{Y}(s_f) = \{\mathbf{y}(0), \mathbf{y}(s_1), \ldots, \mathbf{y}(s_f)\}$. The probability we wish to determine is then written as $P\left(\mathbf{X}(s_f), \mathbf{p} \,\middle|\, \mathbf{Y}(s_f), \mathbf{F}(\cdot)\right)$. Using this notation we can determine the expectation value of some quantity $G(\mathbf{X}(s_f), \mathbf{p})$ as

$$\langle G(\mathbf{X}(s_f), \mathbf{p}) \rangle = \int G\left(X(s_f), \mathbf{p}\right) P\left(\mathbf{X}(s_f), \mathbf{p} \,\middle|\, \mathbf{Y}(s_f), \mathbf{F}(\cdot)\right) d\mathbf{X} d\mathbf{p} \tag{1.5}$$

Estimating the probability and solving the integral in Eq. (1.5) for $G(\mathbf{X}(T_s), \mathbf{p}) = \{\mathbf{X}(T_s), \mathbf{p}\}$ would give an estimate for the full state of the system and all the parameters of motion. Subsequently integrating from the end of the estimation window forward using the equations of motion in Eq. (1.1) will provide an estimated prediction which can be used to validate the parameter and state estimations through comparison of the predicted time evolution of the system to the data.

1.2 Estimating the Probability Distribution

Solving the integral in Eq. (1.5) requires an expression for the probability of the state and parameter values given the data and the model. A summary for the derivation of this expression will be presented here [1]. We begin by making several simplifying assumptions

Assumption 1 The system is Markov: meaning the state at a future time point depends only on the state at the current time point

Assumption 2 Measurement error is additive and there is no correlation between errors in measuring different quantities or at varying time points

Assumption 3 Our mathematical model of the system has some finite probability of being correct

We begin by applying simple Bayesian methods

$$P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_f),\mathbf{F}(\cdot)\right)=\frac{P\left(\mathbf{X}(s_f),\mathbf{y}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_{f-1}),\mathbf{F}(\cdot)\right)}{P\left(\mathbf{y}(s_f)\,\middle|\,\mathbf{Y}(s_{f-1})\right)}$$
$$P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_f),\mathbf{F}(\cdot)\right)=\frac{P\left(\mathbf{X}(s_f),\mathbf{y}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_{f-1}),\mathbf{F}(\cdot)\right)P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_{f-1}),\mathbf{F}(\cdot)\right)}{P\left(\mathbf{y}(s_f)\,\middle|\,\mathbf{Y}(s_{f-1})\right)P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_{f-1}),\mathbf{F}(\cdot)\right)} \tag{1.6}$$

We then note that due to the definition of conditional mutual information

$$CMI(a,b|c)=\log\left[\frac{P(a,b|c)}{P(a|c)P(b|c)}\right] \tag{1.7}$$

we can rewrite the expression in Eq. (1.6) as

$$P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_f),\mathbf{F}(\cdot)\right)=\exp\left[CMI\left(\left\{\mathbf{X}(s_f),\mathbf{p}\right\},\mathbf{y}(s_f)|\mathbf{Y}(s_{f-1})\right)\right]P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_{f-1}),\mathbf{F}(\cdot)\right) \tag{1.8}$$

$$P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_f),\mathbf{F}(\cdot)\right)=\exp\left[CMI\left(\left\{\mathbf{X}(s_f),\mathbf{p}\right\},\mathbf{y}(s_f)|\mathbf{Y}(s_{f-1})\right)\right]$$
$$\cdot P\left(\mathbf{x}(s_f)\,\middle|\,\mathbf{X}(s_{f-1}),\mathbf{p},\mathbf{F}(\cdot)\right)P\left(\mathbf{X}(s_{f-1}),\mathbf{p}\,\middle|\,\mathbf{Y}(s_{f-1}),\mathbf{F}(\cdot)\right) \tag{1.9}$$

$$P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_f),\mathbf{F}(\cdot)\right)=\exp\left[CMI\left(\left\{\mathbf{X}(s_f),\mathbf{p}\right\},\mathbf{y}(s_f)|\mathbf{Y}(s_{f-1})\right)\right]$$
$$\cdot P\left(\mathbf{x}(s_f)\,\middle|\,\mathbf{x}(s_{f-1}),\mathbf{p},\mathbf{F}(\cdot)\right)P\left(\mathbf{X}(s_{f-1}),\mathbf{p}\,\middle|\,\mathbf{Y}(s_{f-1}),\mathbf{F}(\cdot)\right) \tag{1.10}$$

The last step in the above derivation comes from taking Assumption 2. The final term on the right hand side in Eq. (1.10) is identical in form to the probability on the left hand side, but with the measured and estimated paths, $\mathbf{X}$ and $\mathbf{Y}$, containing one fewer time point. We can repeat this process until the $\mathbf{X}$ and $\mathbf{Y}$ sets in the final term contain only $\mathbf{x}(0)$ and $\mathbf{y}(0)$

$$P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_f),\mathbf{F}(\cdot)\right) = P(\mathbf{x}(0),\mathbf{p})\prod_{i=1}^{f}\Bigg\{\exp\left[CMI\left(\{X(s_i),\mathbf{p}\},\mathbf{y}(s_i)|\mathbf{Y}(s_{i-1})\right)\right] \cdot P\left(\mathbf{x}(s_i)\,|\mathbf{x}(s_{i-1}),\mathbf{p},\mathbf{F}(\cdot)\right)\Bigg\} \tag{1.11}$$

In evaluating this expression we make one additional assumption: that we know nothing about the initial condition. This leads to the probability of the initial condition being flat and a constant value. Any multiplicative constants will be divided out of the expression for the expectation value due to the normalization. Due to this, we can simplify Eq. (1.11) into

$$P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_f),\mathbf{F}(\cdot)\right) = C\prod_{i=1}^{f}\Bigg\{\exp\left[CMI\left(\{\mathbf{X}(s_i),\mathbf{p}\},\mathbf{y}(s_i)|\mathbf{Y}(s_{i-1})\right)\right] \cdot P\left(\mathbf{x}(s_i)\,|\mathbf{x}(s_{i-1}),\mathbf{p},\mathbf{F}(\cdot)\right)\Bigg\} \tag{1.12}$$

The second term in the product in Eq. (1.12) is the transfer probability, the probability of a state $\mathbf{x}(s_i)$ given the state at the previous time point $\mathbf{x}(s_{i-1})$ and the model used to integrate the trajectory forward in time. If our model was perfectly accurate and deterministic this would be a delta function. We make Assumption 3 and allow for error in the time propagation function, either through missing or incorrect terms and through stochastic error. To account for these errors we must widen the transfer probability through some approximation of a delta function. We have chosen to use the Gaussian approximation for a delta function.

$$P\left(\mathbf{x}(s_i)\,|\mathbf{X}(s_{i-1}),\mathbf{p},\mathbf{F}(\cdot)\right) \propto \exp\left\{-\frac{R_f}{2}\left|\mathbf{x}(s_i)-\mathbf{F}(\mathbf{x}(s_{i-1}))\right|^2\right\} \tag{1.13}$$

The function $|\cdot|^2$ is the vector inner product. The proportionality in Eq. (1.13) is due to the necessity of the probability being normalized. We define R_f as the inverse standard deviation

squared of the error in the model. This determines the width of the transfer probability. If we assume some error in the model such that

$$\mathbf{x}(s_i) = \mathbf{F}(\mathbf{x}(s_{i-1})) + \boldsymbol{\eta}_f(s_i) \tag{1.14}$$

where $\boldsymbol{\eta}_f(s_i)$ is a term sampled from a distribution that describes the error in the model estimate. R_f is then the inverse of the variance of $\boldsymbol{\eta}_f(s)$. Note that when R_f is zero the error in the model is infinite and thus there is no information in the time series and the transfer probability is uniform. Alternatively, when R_f is infinite the error in the model is zero and the transfer probability is a delta function. When R_f is infinite the transfer probability approaches one in which the model dynamics are strongly constrained, a situation that has been addressed in previous work in DA [27].

All that is left is to determine an expression for the conditional mutual information in Eq. (1.11) before we are able to begin solving for the expectation values of various quantities of interest. This term effectively describes the difference between the estimated states and the measured states. Making Assumption 1, the conditional mutual information takes the form [1]:

$$\exp\left[CMI\left(\left\{X(s_f),\mathbf{p}\right\},\mathbf{y}(s_f)|\mathbf{Y}(s_{f-1})\right)\right] = C\exp\left\{-\frac{R_m}{2}\left|\mathbf{h}(\mathbf{x}(s_i)) - \mathbf{y}\right|^2\right\} \tag{1.15}$$

Here we define R_m as the inverse of the variance of $\boldsymbol{\eta}_m$, first defined in Eq. (1.4). Putting together Eq. (1.13), Eq. (1.15), and Eq. (1.12) we get a new expression for the probability of a path given the data and the model

$$\begin{aligned} P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_f),\mathbf{F}(\cdot)\right) = C\exp\Bigg\{-\Bigg(&\frac{R_f}{2}\sum_{i=1}^{f}\left|\mathbf{x}(s_i) - \mathbf{F}(\mathbf{x}(s_{i-1}))\right|^2 \\ &+\frac{R_m}{2}\sum_{i=0}^{f}\left|\mathbf{h}(\mathbf{x}(s_i)) - \mathbf{y}(s_i)\right|^2\Bigg)\Bigg\} \end{aligned} \tag{1.16}$$

Many methods of DA, such as various modifications of Kalman filtering, typically use *ad hoc* definitions of variable correlations in the formulation of the cost function, the term following the negative sign in the exponent of Eq. (1.16). Though informed by the system and dynamics, these correlation matrices are forced to certain values [16, 39]. Additionally traditional methods assume the value of R_f to be infinite leading to a cost function with strong constraints [27]. Any stochastic or theoretical mistakes in the model would render such a cost function inappropriate for the system of study. In contrast, the methods in this thesis define the variable correlations in the cost function through the model dynamics which specify both spatial and temporal coupling of states and allows for errors in the mathematical model of the system.

1.3 Calculating Expectation Values

Calculating the expectation value of some function $G(\mathbf{X}(s_f))$ involves calculating the integral in Eq. (1.5) using the probability defined in Eq. (1.16). Systems in which the computational methods descussed in this thesis are useful and necessesary are often nonlinear and highly coupled. Integrating over the probability in such systems is often very difficult to do directly as the surface over which it is done is nonlinear and may be scale invariant [12]. As an example, below is a 2-dimensional slice of $P(\mathbf{X}(s_f), \mathbf{p}|\mathbf{Y}(s_f), \mathbf{F}(\cdot))$ for an 11-dimensional Lorenz '96 [28] system in which five dimensions have been 'measured'.

For a probability surface as in Figure (1.1) with many local maxima and irregular structure direct integration over the surface is difficult. There exist many methods for approximating complicated integrals like Eq. (1.5) over non-convex surfaces. One such method is the Laplace method [23, 24] which entails approximating the expectation value of a function with the weighted sum of the function calculated along the paths located at the probability maxima. The weights for each component of this sum are determined by the probability of the path. The path $\mathbf{X}^0$ associated with the largest maximum of the probability $P(\mathbf{X}(s_f), \mathbf{p}|\mathbf{Y}(s_f), \mathbf{F}(\cdot))$ contributes to

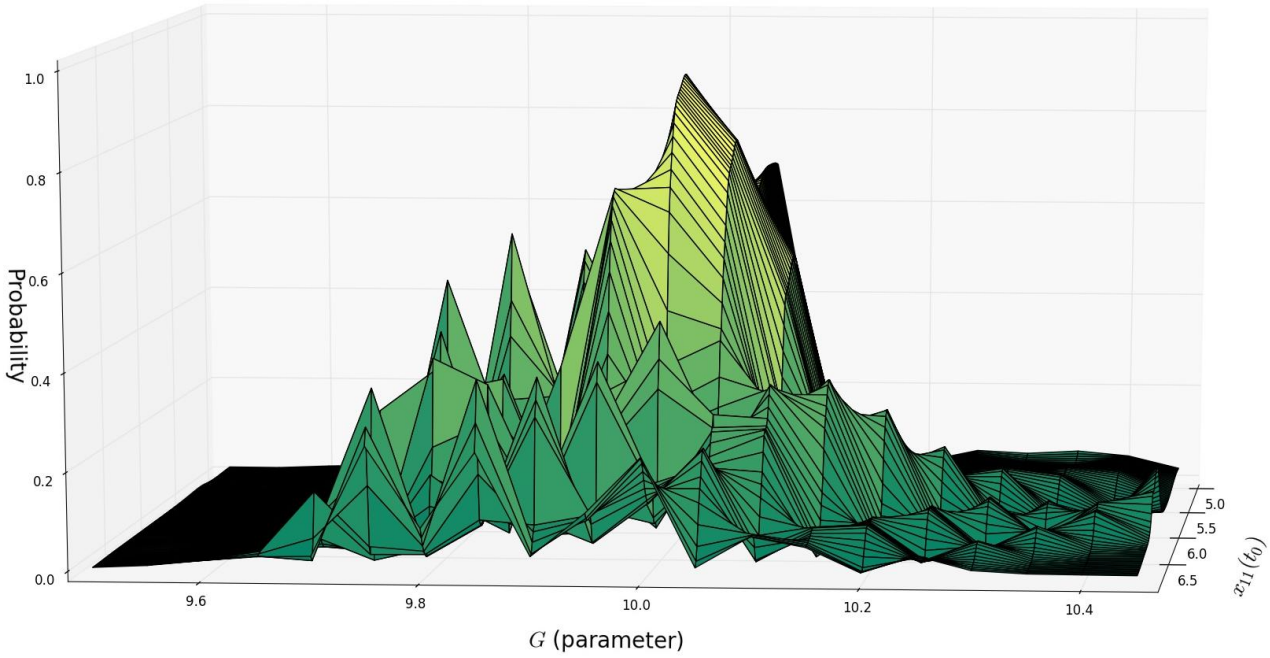


Figure 1.1: 2-dimensional probability slice of $D = 11$, $L = 2$ chaotic Lorenz '96 system. Slice is taken over the values of the initial condition of an unmeasured variable and the constant forcing term. Probability Surface is riddled with local maxima making it difficult to directly integrate over this surface

the expected value integral by a factor approximately equal to $\frac{P(\mathbf{X}^0(s_f),\mathbf{p}|\mathbf{Y}(s_f),\mathbf{F}(\cdot))}{P(\mathbf{X}^1(s_f),\mathbf{p}|\mathbf{Y}(s_f),\mathbf{F}(\cdot))}$ more than the path $\mathbf{X}^1$ associated with the second largest maximum.

The use of the Laplace method requires that the width of the peaks in probability distribution are small so that the paths located at probability maximum contribute significantly more to the integral than the paths along the decay of these maxima. We make a further simplification by assuming that there is a single peak in the probability density function which is significantly larger than all other maxima. By assuming that the probability density function contains a maximum that is significantly larger than all others and that the corresponding peak in the probability density function is sufficiently narrow, we can approximate the expectation value in Eq. (1.5) with the most probable value. In other words

$$\langle G(\mathbf{X}(s_f),\mathbf{p})\rangle \approx G\left(\underset{\mathbf{X}(s_f),\mathbf{p}}{\operatorname{argmax}}\left[P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_f),\mathbf{F}(\cdot)\right)\right]\right) \tag{1.17}$$

Here the output of argmax is the $\{\mathbf{X}(s_f),\mathbf{p}\}$ path and parameter values that have the highest probability. To find the most probably state, we must maximize the probability shown in Eq. (1.16). This is difficult for two reasons:

1. The probability is often very nonlinear and has multiple maxima. In fact the maximization of this probability is often an NP-complete problem [33]

2. Without knowing the underlying dynamics exactly, it is difficult to estimate the error in the mathematical model used and thus the appropriate value of R_f

There exists an optimization method to avoid these problems, called Variational Annealing (VA) [37, 38]. To apply this method we first redefine the probability in terms of an Action such that

$$P\left(\mathbf{X}(s_f),\mathbf{p}\,\middle|\,\mathbf{Y}(s_f),\mathbf{F}(\cdot)\right) = e^{-A\left(\mathbf{X}(s_f),\mathbf{p}\middle|\mathbf{Y}(s_f),\mathbf{F}(\cdot)\right)} \tag{1.18}$$

Due to this relationship, maximizing the probability density function is equivalent to minimizing the action. We drop the dependence of the action on $\mathbf{Y}(s_f)$ and $\mathbf{F}(\cdot)$ from this point forward for notational simplicity. This leads to an equation for the action in the form

$$A\left(\mathbf{X}(s_f),\mathbf{p}\right) = \frac{R_f}{2}\sum_{i=1}^{f}\left|\mathbf{x}(s_i)-\mathbf{F}(\mathbf{x}(s_{i-1}))\right|^2 + \frac{R_m}{2}\sum_{i=0}^{f}\left|\mathbf{h}(\mathbf{x}(s_i))-\mathbf{y}(s_i)\right|^2 \tag{1.19}$$

We can generalize the Action in Eq. (1.19) back to continuous time by taking the limit as Δs approaches zero and the number of total time points approaches infinity in such a way that the total time window remains constant:

$$A_{continuous}\left(\mathbf{X}(s_f),\mathbf{p}\right)=\int_{s_0}^{s_f}dw\left[\frac{R_f}{2}\left|\dot{\mathbf{x}}(w)-\mathbf{f}(\mathbf{x}(w))\right|^2+\frac{R_m}{2}\left|\mathbf{h}(\mathbf{x}(w))-\mathbf{y}(w)\right|^2\right] \tag{1.20}$$

The continuous time Action in Eq. (1.20) is analogous to the traditional action in classical physics with the Lagrangian given by:

$$\mathcal{L}(\mathbf{x}(w),\dot{\mathbf{x}}(w),w)=\left[\frac{R_f}{2}\left|\dot{\mathbf{x}}(w)-\mathbf{f}(\mathbf{x}(w))\right|^2+\frac{R_m}{2}\left|\mathbf{h}(\mathbf{x}(w))-\mathbf{y}(w)\right|^2\right] \tag{1.21}$$

We can determine the path $\mathbf{X}^0$ which minimizes the action in Eq. (1.20) through the solution of the Euler-Lagrange (EL) equations. The solution of the EL equations will thus determine the most likely path and approximate the mean path given the recorded data. Solving the EL equations directly is impossible except in special circumstances. Due to this constraint, we apply numerical methods of optimization to determine an estimate for $\mathbf{X}^0$.

1.3.1 Variational Annealing

Variational Annealing is a method meant to alleviate the problems discussed in the previous section with numerically optimizing a complex and nonlinear surface. The premise of VA is to slowly and adiabatically enforce the model constraints given by the function defined in Eq. (1.2) [37]. The method for this slow enforcement of the model constraints is through the gradual increase of R_f. As previously mentioned, when R_f is zero the transfer probability from one time point to another is uniform, thus the probability density function in state and probability space is flat in all unmeasured directions and is Gaussian centered at the measured value with width equal to the noise in the direction of the measured variables. The following steps are then taken

1. Set $R_f = 0$

2. Using random initial guesses for the path and parameter values minimize the action

3. Set $R_f = R_{f,0}\alpha^{\beta}$ with β defined as the iteration number and with some sufficiently small value of $R_{f,0}$ and $\alpha > 1$. Define a new action surface with these values

4. Using the values for the path and parameter found in the previous minimization of the action as the initial points, minimize the action

5. repeat Step (3) and Step (4) until the value of the minimized action becomes independent of R_f

In Step (3) the value of α must be greater than one, but small enough that the surface of the action changes smoothly from one iteration to the next. The smooth transformation allows the minimization to 'track' the evolution of the minimum of the action as R_f increases. Once the location in path and parameter space is found that exactly fits the model of the system, the model error term in the action becomes zero. At this point the value of R_f has no effect on the action and the 'ideal' path has been discovered.

When the number of measured variables is much smaller than the number of total states, the initial action in Step (2) is flat in most directions. This allows for a wide range of initial search locations leading to a likelihood that any single optimization routine will be 'stuck' in a local minimum even with the annealing method. To combat this problem, several (typically around the order of 100) random initial paths are chosen as starting points and each is passed through the annealing routine. Depending on the structure of the action, this leads to several action 'levels' at high R_f values. When the lowest level is significantly smaller than all the others, this single level may be used to approximate the expectation value of any desired function $G(\mathbf{X}(s_f), \mathbf{p})$.

1.4 Shortcomings

Though variational annealing is a powerful method, it has several shortcomings due to the fact that it utilizes optimization of a surface. Results of optimization, by definition, do not contain any information as to the structure of the surface being optimized such as the width of peaks and any higher moments. Optimization of any form uses the Laplace approximation when estimating the expectation value of the probability density and assumes and is only accurate in the cases where there is a rapidly decaying probability maxima. Without prior knowledge of the structure of the distribution, it is not possible to determine if the Laplace approximation is appropriate. Additionally, due to the lack of information with regards to higher moments, optimization gives no way to determine error bars or confidence intervals for estimates made through this method. When applied to experimental data, such error bars are crucial to understanding the system of study and assessing the quality of the estimates made.

We investigate an extension to VA that allows for informed exploration of the structure of $P(\mathbf{X}|\mathbf{Y})$. In multimodal systems, we are interested only in the structure of $P(\mathbf{X}|\mathbf{Y})$ near the largest maxima as those regions contribute most significantly to the expected values of the statistical quantities of interest. We develop and describe here a method through which the results of a VA analysis are used to inform the boundaries of a Metropolis-Hastings Monte Carlo [31] search in path space. The Monte Carlo search is constrained to regions of interest, the regions of high likelihood determined through optimization of the probability distribution using VA. This targeted search reduces the computational time necessary to accurately estimate the structure of $P(\mathbf{X}|\mathbf{Y})$ at high probability regions of state and parameter space by avoiding regions of low probability that do not significantly contribute to the integrals in Eq. (1.5). Combining a variational method with a constrained random search provides the benefits of both the faster variational method and the more thorough and informative Monte Carlo search.

1.5 Overview of Thesis

This thesis develops a computational method in which information about the structure and higher moments of the probability distribution can be extracted from the measured data and the model. A previous method, the Maximum Likelihood Ensemble Filter method (MLEF) [39] attempts to provide such information as to the structure and higher moments of the distribution though does not address error introduced through inaccuracies in the model. MLEF enforces strong model constraints that may lead to a search surface punctuated by many local minima and may constrain the search to a region of state space which does not properly describe the data due to stochastic or theoretical errors in the model constraints. The strategic Monte Carlo (SMC) method presented and applied in this thesis differs from MLEF in part by allowing for error in the model as first addressed in work on VA [1, 38].

The second chapter of this thesis develops the method underlying SMC and demonstrates an application using data simulated from a chaotic Lorenz '96 model. This model is chosen due to its flexibility: the Lorenz '96 system can easily be generalized to any number of degrees of freedom and can exist both in an oscillatory or chaotic regime depending on the value of the constant forcing parameter.

In the third chapter SMC is applied to a simulated neuron experiment. SMC's ability to determine the form of the probability distribution is demonstrated and assumptions about the structure of the probability of states and parameter values conditioned on the data made in prior work [35, 18, 5] are validated through the results here.

The fourth chapter discusses a generalization of DA methods to the field of Machine Learning (ML) through the use of Neural Networks (NN). This chapter addresses the equivalence between the optimization problem outlined in the Introduction and the one underlying ML. The use of the weakly constrained Action, defined in the introduction, is proposed as a tool for future ML research.

Chapter 2

Strategic Monte Carlo

In the previous chapter we described a method, Variational Annealing (VA), for using the hyper-parameter R_f, or the model precision, as a tool for identifying the smallest minima of the action. When the smallest of these minima is much smaller than the magnitude of the next minimum, it dominates the expected value integral Eq. (1.5). When this occurs, Laplace's approximation [23, 24] for the expectation value of a function over the probability density is valid. Our focus in this chapter is on methods to sample $P(\mathbf{X}, \mathbf{p}|\mathbf{Y})$ when the conditions in which optimization is an appropriate method are not met. In such cases the structure of the probability distribution must be taken into consideration when solving the integral in Eq. (1.5).

We develop a method, Strategic Monte Carlo (SMC), to combine optimization and random search methods to acheive better estimates of a complex probability density fuction while decreasing the computational time needed to search this space. SMC is a process in which a random search is conducted over the probability density function and is constrained through importance sampling methods to regions in path space of high likelihood corresponding to the locations of the local Action minimima found through VA.

This chapter will discuss the mathematics of SMC and demonstrate its application to the estimation of states and parameters in simulated chaotic 11-dimensional Lorenz '96 system to

test its effectiveness.

2.1 Background

We have written the conditional probability density function $P(\mathbf{X},\mathbf{p}|\mathbf{Y})$ as proportional to

$$P(\mathbf{X},\mathbf{p}|\mathbf{Y}) \propto e^{-A(\mathbf{X},\mathbf{p})}$$
$$A(\mathbf{X},\mathbf{p}) = \sum_{t=0}^{t_F}\left[\sum_{a=1}^{L}\frac{R_m(a,t)}{2}\left(x_a(t)-y_a(t)\right)^2+\sum_{a=1}^{D}\frac{R_f(a)}{2}\left(x_a(t+1)-F_a(\mathbf{x}(t),\mathbf{p})\right)^2\right], \qquad (2.1)$$

in which the independent variable, s, has been relabeled t to explicitly denote time. $R_m(a,t)$ is the (diagonal) precision matrix for the Gaussian noise in the measurements, and $R_f(a)$ is the (diagonal) precision matrix for the Gaussian error in our model $\mathbf{x}(t+1)=\mathbf{f}(\mathbf{x}(t),\mathbf{p})$. We no allow both of the precision matrices to vary as a function of state, a, and measured time, t.

The first term in $A(\mathbf{X},\mathbf{p})$ in Eq. (2.1) reflects a Gaussian error in the measurements, while the second term describes the model error, also taken to be Gaussian. $P(\mathbf{X},\mathbf{p}|\mathbf{Y})$ is not Gaussian as $\mathbf{F}(\mathbf{x},\mathbf{p})$ is nonlinear.

While other choices of error distribution are certainly possible, this expression of measurement error and model error is analyzed so frequently we call this the 'standard model' for statistical data assimilation.

As mentioned in the introduction, when $R_f \to \infty$ the model is satisfied exactly (becomes deterministic). Alternatively, when $R_f \to 0$, the standard deviation of model error is infinite and nothing can be learned about the time evolution of the system. At $R_f = 0$ the probability distribution is a Gaussian with width given by the measurement noise in the measured state variable directions and flat in the direction of unmeasured variables. When there is zero precision in the model, information is not transfered from measurements to unmeasured model states.

If the discrete time model dynamics defined by the discretization $\mathbf{F}(\mathbf{x},\mathbf{p})$ of the function $\mathbf{f}(\mathbf{x},\mathbf{p})$ perfectly describes the system and all state variables are measured, then the probability will

approach an approximation of a delta function with width given by the measurement accuracy. As the number of measurements decreases, the probability will broaden in the unmeasured directions and become more complex and multimodal depending on the structure of the equations of motion and the form of the coupling between measured and unmeasured degrees of freedom in $\mathbf{f}(\mathbf{x},\mathbf{p})$. While the value of R_f approaches infinity and the model is deterministic, the transfer probability from a time t to a time $t+1$ will remain a delta function and the probability of the path will depend entirely on the estimates of the parameters and initial conditions of the state variables at time t_0. With a finite R_f value the model is non-deterministic and the state at one time point does not lead directly to the state at the next time point. The transfer probability from one time to the next widens and independent estimates of the state variables at all time points are required to accurately approximate the integral $\langle G(\mathbf{X},\mathbf{p})\rangle = \int d\mathbf{X} d\mathbf{p} G(\mathbf{X},\mathbf{p}) P(\mathbf{X},\mathbf{p}|\mathbf{Y})$.

2.2 Monte Carlo

A Monte Carlo procedure is a type of random walk search. In these searches, a 'walker' moves through path space with the probability landscape guiding the direction it steps. Over a long enough walk, the amount of time spent in a region of state space becomes proportional to the probability of those states. When sampling from a distribution using a Metropolis-Hastings algorithm, first developed by Metropolis, et al [31], the search procedure consists of two steps:

1. start at some point in path space $\{\mathbf{X},\mathbf{p}\}$; choose, or sample, a new point $\{\mathbf{X}',\mathbf{p}'\}$ in path space selecting from $P_{sample}(\mathbf{X}',\mathbf{p}')$

2. decide whether or not to **accept** this new point and to 'step' to it with $P_{accept}(\mathbf{X}',\mathbf{p}')$, or **choose to reject** this new point, and repeat the whole procedure starting with the same initial point in $\{\mathbf{X},\mathbf{p}\}$.

It is important that the distribution from which the 'new' point is sampled overlaps strongly

with the search distribution. Intuitively this is because it is impossible to search a region that is unsampled. Mathematically, this is because the distribution that is searched is equal to the product of the sampling distribution and the acceptance/rejection distribution. If the sampling distribution is zero where sampling occurs, the search will result in a zero probability regardless of the acceptance/rejection probability since

$$P_{output}(\mathbf{X},\mathbf{p}|\mathbf{Y}) = P_{sample}(\mathbf{X},\mathbf{p}|\mathbf{Y}) \cdot P_{accept}(\mathbf{X},\mathbf{p}|\mathbf{Y}). \tag{2.2}$$

For notational simplicity we drop the dependence on the data, $\mathbf{Y}$ in the following discussion.

We initialize our walker at location, $\{\mathbf{X},\mathbf{p}\}$, and sample a location, $\{\mathbf{X}',\mathbf{p}'\}$. The probabilities of sampling the next location and of accepting it are conditioned on the previous walker location. We use the property of detailed balance which states that

$$\frac{P(\mathbf{X}',\mathbf{p}')}{P(\mathbf{X},\mathbf{p})} = \frac{P(\mathbf{X}',\mathbf{p}'|\mathbf{X},\mathbf{p})}{P(\mathbf{X},\mathbf{p}|\mathbf{X}',\mathbf{p}')} \tag{2.3}$$

to connect the absolute probability of a state with a conditional probability of a state of the walker given its previous state.

From here we have

$$\frac{P(\mathbf{X}',\mathbf{p}')}{P(\mathbf{X},\mathbf{p})} = \frac{P_{sample}(\mathbf{X}',\mathbf{p}'|\mathbf{X},\mathbf{p})P_{accept}(\mathbf{X}',\mathbf{p}'|\mathbf{X},\mathbf{p})}{P_{sample}(\mathbf{X},\mathbf{p}|\mathbf{X}',\mathbf{p}')P_{accept}(\mathbf{X},\mathbf{p}|\mathbf{X}',\mathbf{p}')} \tag{2.4}$$

A common, and easy to sample, choice for these probabilities is $P_{sample}(\mathbf{X}',\mathbf{p}'|\mathbf{X},\mathbf{p})$ selected as a Gaussian centered at $\{\mathbf{X},\mathbf{p}\}$ with some standard deviation given by the desired 'step size', where P_{sample} is a function of $|\mathbf{X}-\mathbf{X}'|$.

A sampling probability that is symmetric, like the commonly used Gaussian distribution, transforms Eq. (2.4) to the simplified expression:

$$\frac{P(\mathbf{X}',\mathbf{p}')}{P(\mathbf{X},\mathbf{p})} = \frac{P_{accept}(\mathbf{X}',\mathbf{p}'|\mathbf{X},\mathbf{p})}{P_{accept}(\mathbf{X},\mathbf{p}|\mathbf{X}',\mathbf{p}')} \tag{2.5}$$

We then can set

$$P_{accept}(\mathbf{X}',\mathbf{p}'|\mathbf{X},\mathbf{p}) = \min\left[1, \frac{P(\mathbf{X}',\mathbf{p}')}{P(\mathbf{X},\mathbf{p})}\right] \tag{2.6}$$

as a possible acceptance probability. These choices uniformly sample the state space of the problem. The search is allowed to run for a sufficiently long time that the location of the walker becomes decorrelated from the initial location. The probability of a location is subsequently determined by the fraction of time the walker spent at this location.

This method was applied by Kostuk, et al [21] using a definition of the probability distribution function containing a model error term, often ignored in other Monte Carlo or random walk estimation routines. As previously discussed, when model error is accounted for in the probability distribution function the state at each time point in the trajectory must be separately searched over because the transition from one time point to the next is not deterministic. This increases the dimension of search by a multiplicative factor equal to the number of time points used in the analysis. The number of random walk steps necessary to estimate a target distribution increases with the dimension of the space explored [30]. A multiplicative increase in the dimension of the problem equal to the number of time points in the data will cause the necessary computational time for a random walk procedure to grow to unfeasible lengths. To decrease the computational time necessary to make parameter and state estimates through a Monte Carlo search of the probability distribution we introduce a method to constrain and guide the search to regions of high likelihood.

2.2.1 Importance Sampling Monte Carlo

If some structure of the desired distribution is known, it is possible to explore only the higher probability regions of state space using the methods of Importance Sampling [8, 13]. A sampling probability $P_{sample}(\mathbf{X}', \mathbf{p}'|\mathbf{X}, \mathbf{p})$ is chosen such that it depends explicitly on $\mathbf{X}$ as well as the distance between $\mathbf{X}$ and $\mathbf{X}'$. This sampling probability is often smoother and easier to sample than the desired distribution, but it gives a rough approximation of the target distribution. For example,

$$P_{sample}(\mathbf{X}'|\mathbf{X}) = P_{bias}(\mathbf{X}', \mathbf{p})P_{step}(\mathbf{X}', \mathbf{p}'|\mathbf{X}, \mathbf{p}) \tag{2.7}$$

Where P_{step} is the previous P_{sample}, a Gaussian with standard deviation given by desired step size and symmetric in $\mathbf{X}$ and $\mathbf{X}'$. The acceptance probability is left unchanged. Consequently, the output distribution is given by

$$P_{output}(\mathbf{X}', \mathbf{p}') = P(\mathbf{X}'|\mathbf{p}')P_{bias}(\mathbf{X}', \mathbf{p}') \tag{2.8}$$

To estimate the desired distribution of $P(\mathbf{X}', \mathbf{p}')$, the output of the search algorithm is multiplied by a weight given by

$$W(\mathbf{X}', \mathbf{p}') = \frac{1}{P_{bias}(\mathbf{X}', \mathbf{p}'|\mathbf{X}, \mathbf{p})}. \tag{2.9}$$

This weight removes any bias introduced into the outcome by sampling certain regions more often than others. To thoroughly explore all regions of interest or high probability, it is necessary that $P_{bias}(\mathbf{X}', \mathbf{p}')$ overlap strongly with the high likelihood regions of $P(\mathbf{X}', \mathbf{p}')$.

2.3 Methods

Though optimization of the probability distribution through the VA optimization of the action provides a good estimate for the location of many of the maxima of the conditional probability density function, in many cases, such as those in which the probability is asymmetric or has fat tails, the location in state space of the maxima is insufficient to estimate expectation values and higher moments for a distribution. An approximation of the structure of the probability is needed. To address this need we apply the use of Importance Sampling discussed in the previous section to the probability surface.

We implement an informed random walk that is constrained to regions of interest in state space by the results of the previous VA analysis. In other words we execute an importance sampling Monte Carlo in which the sampling distribution is informed by the location of the local maxima in the probability as determined by the variational annealing method. The results of VA optimization of the distribution is used to determine an appropriate bias probability, P_{bias}, such that the bias probability is approximately centered at the maximum probability region of the desired probability density and wide enough to encompass all regions of interest.

As the search space, and thus the resulting probability density function, is extremely high dimensional, it is useful for visualization of the results to project the estimated distribution into a lower dimensional space. Because parameters are constants in motion and because they completely describe the model dynamics of the system, when the functional form of $\mathbf{x}(n+1) = \mathbf{F}(\mathbf{x}(n), \mathbf{p})$ is given, we project the probability density function to the smaller space of the parameter vector.

$$P(\mathbf{p}') = \int P_{output}(\mathbf{X}', \mathbf{p}') W(\mathbf{X}', \mathbf{p}') d\mathbf{X}' \tag{2.10}$$

Due to the discrete nature of the random walk search and the integral in Eq. (2.10) is estimated as a sum over all accepted steps in the walker's path.

$$P(\mathbf{p}') = \sum_{j=0}^{N_{accepted}} \frac{1}{N_{accepted}} W(\mathbf{X}'_j, \mathbf{p}') \tag{2.11}$$

Where $N_{accepted}$ is the number of total accepted paths. And the probability and $P_{output}(\mathbf{X}', \mathbf{p}')$ is approximated by the frequency of the path $\{\mathbf{X}', \mathbf{p}'\}$ being accepted.

In this chapter we will apply the method developed in this thesis to the Lorenz '96 system in which there is only one parameter. With one parameter this sum produces a probability distribution in one dimension which can be easily visualized.

2.3.1 Informing the Random Walk through VA

Information on the location of local maxima in the probability distribution function determined by VA can be used to inform the bias probability described in the previous section. We set the center of $P_{bias}(\mathbf{X}', \mathbf{p}')$ at the location of the highest probability path found by the VA method. As a first order approximation, we set the bias probability to be an uncorrelated multivariate Gaussian in all dimensions of $\{\mathbf{X}', \mathbf{p}\}$. In each dimension the standard deviation of the Gaussian is set to be four times the distance between the location in state space of the largest and smallest local maxima of the probability found by the VA method.

In other words, define a path $\{\mathbf{X}_0, \mathbf{p}_0\}$ which sets the location of the largest local maximum and $\{\mathbf{X}_1, \mathbf{p}_1\}$ that sets the smallest local maximum found by VA. Recall that $\mathbf{X} \equiv \{\mathbf{x}(t_0), \mathbf{x}(t_1), \ldots, \mathbf{x}(t_F)\}$. If only one local maximum is found at the chosen R_f, or model precision, value we set $\{\mathbf{X}_1, \mathbf{p}_1\}$ to be equal to a smaller local probability maximum found at a previous set in the annealing procedure. We define the distance, D_s, between these paths as

$$\begin{aligned} D_s =& (|\mathbf{X}_0 - \mathbf{X}_1|, |\mathbf{p}_0 - \mathbf{p}_1|) \\ D_s =& \Big(\{|\mathbf{x}_0(t_0) - \mathbf{x}_1(t_0)|, |\mathbf{x}_0(t_1) - \mathbf{x}_1(t_1)|, \\ & \ldots, |\mathbf{x}_0(t_F) - \mathbf{x}_1(t_F)|\}, |\mathbf{p}_0 - \mathbf{p}_1| \big) \end{aligned}$$

Where $|\mathbf{A}| \equiv |A_1| \cdot \hat{e}_1 + |A_2| \cdot \hat{e}_2 + \ldots$.

The dimension of each $\mathbf{x}(t)$ is D, the number of dynamical variables. To allow for the same range of search at every time point for a given dynamical variable, we set the standard deviation for the bias probability of the i'th element of $\mathbf{x}$ at any time, t_j, to be equal to four times the largest distance in that component of $\mathbf{x}$. The factor of four is chosen to maintain weak bias in the sampling while allowing all local probability maxima to fall within the well sampled region of $\{\mathbf{X}, \mathbf{p}\}$. Or in other words

$$\sigma^i_{search}(t_j) = \sigma^i_{search} = 4 \max_t \left| x^i_0(t) - x^i_1(t) \right| \tag{2.12}$$

for the search in state space, and

$$\boldsymbol{\sigma}^p_{search} = 4 \left| \mathbf{p}_0 - \mathbf{p}_1 \right| \tag{2.13}$$

for the search in parameter space.

The acceptance probability is defined by using Eq. (2.1) at the current and proposed locations in state space, $\{\mathbf{X}, \mathbf{p}\}$ and $\{\mathbf{X}', \mathbf{p}'\}$, in Eq. (2.6).

This bias probability which contains all the local maxima found by VA within one fourth of a single standard deviation gently constrains the walkers to stay within regions of high probability as found by VA while still given freedom to explore the structure of the tails of the probability distribution.

2.4 Results

We demonstrate the Monte Carlo Extension to VA in the following section. Testing of this method is done on the Lorenz '96 system. The system itself can be characterized by a set of D coupled differential equations with $D \geq 3$. In addition to the dynamical variables, $x_i(t)$ the

system has a constant forcing term G.

$$\frac{dx_i(t)}{dt} = (x_{i+1}(t) - x_{i-2}(t))x_{i-1}(t) - x_i(t) + G \tag{2.14}$$

The subscripts in $x_i(t)$ are calculated $\mod(D)$. As the magnitude of G grows it moves the behavior of the system from the non-chaotic to chaotic regimes. The transition into chaotic regime occurs at approximately $G = 8$ [19, 28].

2.4.1 Twin Experiments

Twin experiments are experiments run on simulated data. These can be used to test methods, design experiments, or determine various requirements such as noise level or measurement frequency when recording data in a real experiment. Twin experiments are useful because having generated the data we know all of the 'unmeasured' quantities in the system. This allows us to test whether any estimates of unmeasured states or parameters match those not presented to the analysis and thus whether the data presented provides sufficient information regarding the dynamics to fully describe the system.

A mathematical model, identical to the one that best describes the system of study, is used to generate simulated data. 'Noise' is added to the system output to mimic measurement error, with the level of noise matching the known accuracy of the equipment used in a real experiment. A noisy time series of $L \leq D$ variables is presented to the estimation procedure as the only available 'data'. Analysis is then done on the simulated data and results are compared with the known underlying dynamics of the system.

For the work discussed in this chapter, we dealt with a set of twin experiment examples of the Lorenz '96 system. The data for these experiments were collected from a $D = 11$ dimensional example with an estimation window containing 165 time points separated by $\Delta t = 0.025$ units. We ran parameter and state estimations using a varying number L of measured states. And in all

cases we added approximately 3.4% uniformly distributed measurement noise to the 'measured' variables. Our experiments varied the number of measured states to demonstrate the dependence of results on L.

The threshold level of collected data necessary to extract sufficient information through optimization methods to properly estimate all state variables and parameters of this model has been determined to be approximately 40% of the degrees of freedom [20]. The quantity of measured data in the following sections ranges from less than this threshold to above it. Around the threshold there is a qualitative shift in the results.

2.4.2 Estimation on an Eleven Dimensional System

We set the magnitude of the forcing term in this twin experiment to be $G = 10$. Initial states used to generate the data are randomly chosen within the dynamical range of the variables: approximately $[-10, 10]$. The equations of motion Eq. (2.14) were integrated forward using the odeInt package in Python. Uniformly distributed noise with magnitude 17% of the dynamical range of the variables of motion was added following the integration. Below is a time series demonstrating the behavior of the first state, x_0. This time series shown in Figure (2.1) is similar to the time series of all other states.

Analysis was done using 2, 4, 5, and 7 measured states. In all cases the measured states were distributed over the system as symmetrically as possible. For example, in the case of two measured states, we measured $x_0(t)$ and $x_5(t)$. This was done because the coupling in Lorenz '96 is a type of nearest neighbor coupling and clustering the measured states together does not provide sufficient information to make accurate estimates of unmeasured states [20].

Variational Annealing

Figure (2.2) demonstrates the results achieved by applying VA to the Lorenz '96 system with L = 2, 4, 5, and 7 measurements respectively. The action is given by the negative log of

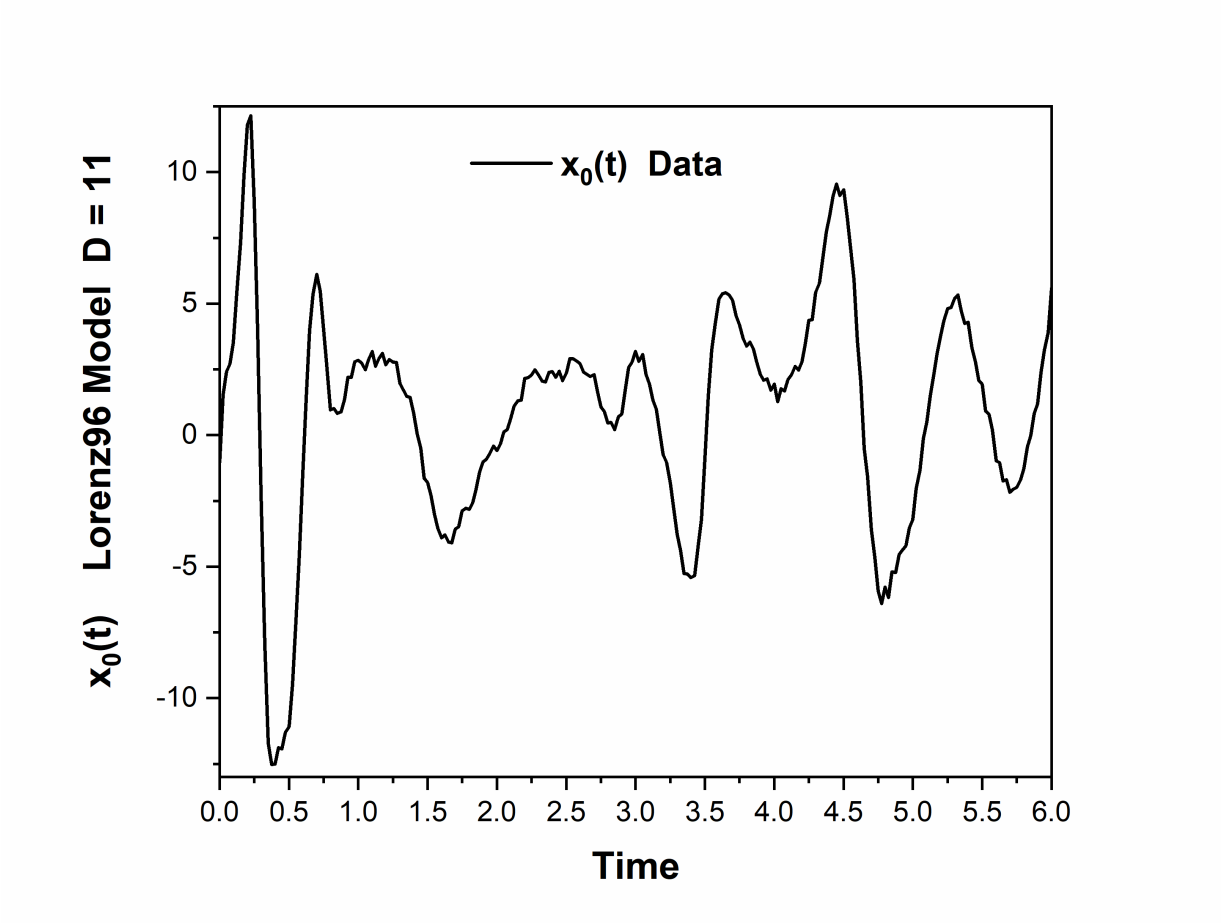


Figure 2.1: Time series data for $x_0(t)$ from the Lorenz '96 equations $D = 11$. The forcing parameter is $G = 10.0$.

the probability, $A(\mathbf{X},\mathbf{p}) = -\log[P(\mathbf{X},\mathbf{p})]$. As $\beta = \log\left[\frac{R_f}{R_m}\right]$ increases, the action is increasingly dominated by the model constraint. The relative magnitude of the R_f and R_m terms in the action is equal to β as shown in Eq. (2.1).

Figure (2.2) demonstrates that as L increases, we see the lowest action/highest probability path increasingly separate from the other local action minima. This implies that the probability maximum is exponentially higher on this high likelihood path and that the estimate for the expectation value of the states and parameter will become increasingly dominated by this path. This increasing separation implies that more measured data allows a more dominant maximum likelihood path than when there are fewer measured states. Intuitively this should be clear.

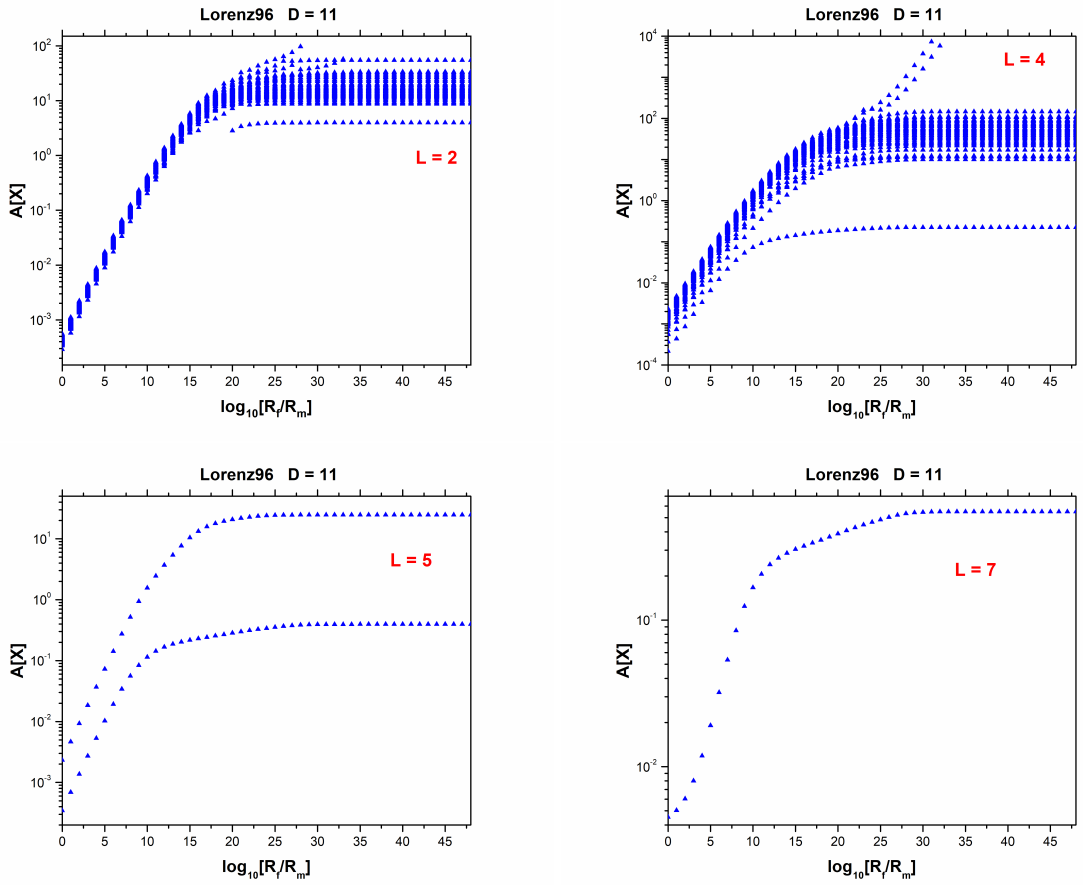


Figure 2.2: Action level plots for the Lorenz 96 model, Eq. (2.14) D = 11, and with L = 2,4,5,7 observed state variables. The number of minima in the standard action Eq. (2.1) decreases as the number of measurements increase. This implies a smoothing of the action $A(\mathbf{X})$ and of the probability density function ($\propto \exp[-A(X)]$) with increased information

The minimization that produces the plots in Figure (2.2) was performed using the open source software, IPOPT [36]. The optimization was carried out using 100 initial conditions with 165 time steps of data for each of the plots shown.

The prediction error was then calculated for every initial condition at every value of R_f used in the annealing procedure. Prediction was made by using Python's odeint and integrating forward $T_{pred} = 100$ time steps from the value of the state $\{\mathbf{x}(t_F), \mathbf{p}\}$. The predicted path was compared to the 'data' using a least squares algorithm:

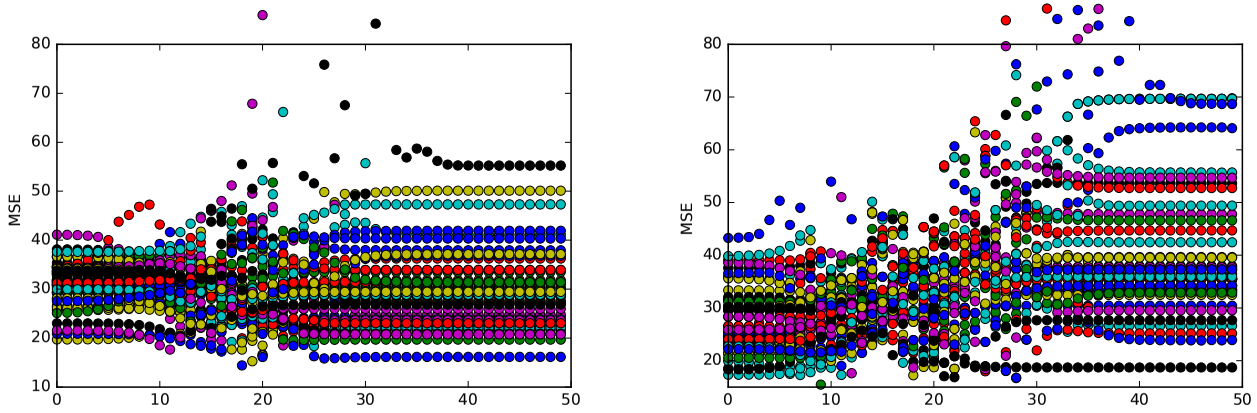
$$MSE = \frac{1}{T_{pred} L} \sum_{t=0}^{T_{pred}} |\mathbf{y}(t_F + t) - \mathbf{x}(t_F + t)|^2 \tag{2.15}$$

The squared term in Eq. (2.15) is the vector inner product. Prediction error, MSE, was averaged over the number of measurements made so a comparison can be made between the error at each L value. The initialization and R_f value that generated the lowest prediction error was used to set the location of the center of the Monte Carlo search in path space and the estimate for the value of R_f (the model precision). The initialization and R_f value that generated the highest prediction error was used to set the width of the sampling probability in path space as discussed in Section 2.3.1.

At any given T_{pred} value, we expect the MSE to depend on the error in estimation of the initial state of the prediction, Δ. The magnitude of Δ is in turn proportional to the value of the action of the corresponding path vector. We expect the distribution of MSE's over all paths to reflect the distribution of action values. We see that this is the case by comparing the plots in Figure (2.2) and Figure (2.3) for large $\beta = \log \frac{R_f}{R_m}$ values in which the model is strongly constrained throughout the state and parameter estimation.

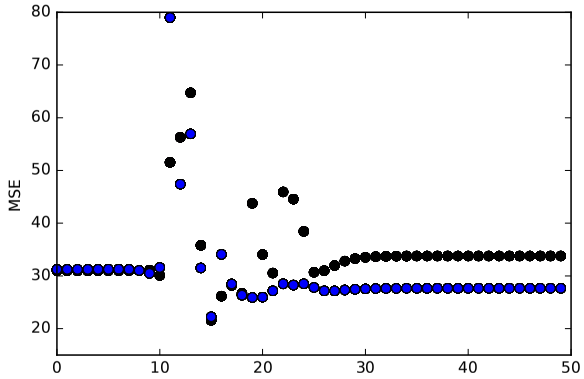
The path producing the lowest action level in Figure (2.2) corresponds to the one producing the lowest prediction error in Figure (2.3). These results validate the choice described in Section 2.3.1 for selecting the lowest MSE path as the center of the Importance Sampling search and as a first order estimate for the highest probability path.

Since we have chosen the value of the forcing term, G, to be within the range in which the Lorenz '96 system is chaotic, we expect large errors in prediction even when the estimates of the parameter and final states are relatively accurate. In chaotic systems the upper bound of the error in a prediction grows exponentially as $\Delta \cdot e^{t\lambda}$ where λ is the largest Lyapunov exponent of the system [10]. In a chaotic system such as this one, the Lyapunov exponent in at least one of the degrees of freedom is positive. This leads to growing error as T_{pred} grows. A finite maximum MSE value, determined by the size of the attractor of the system, is reached at large T_{pred} for the

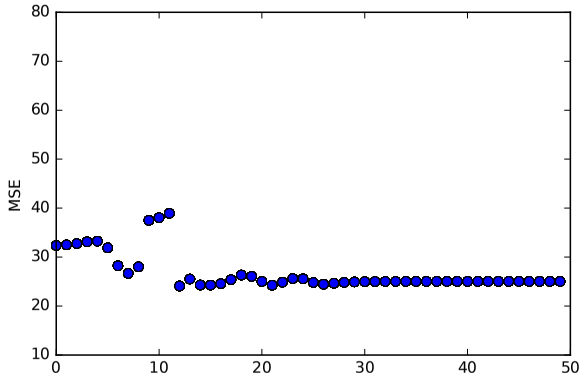


(a) L=2

(b) L=4

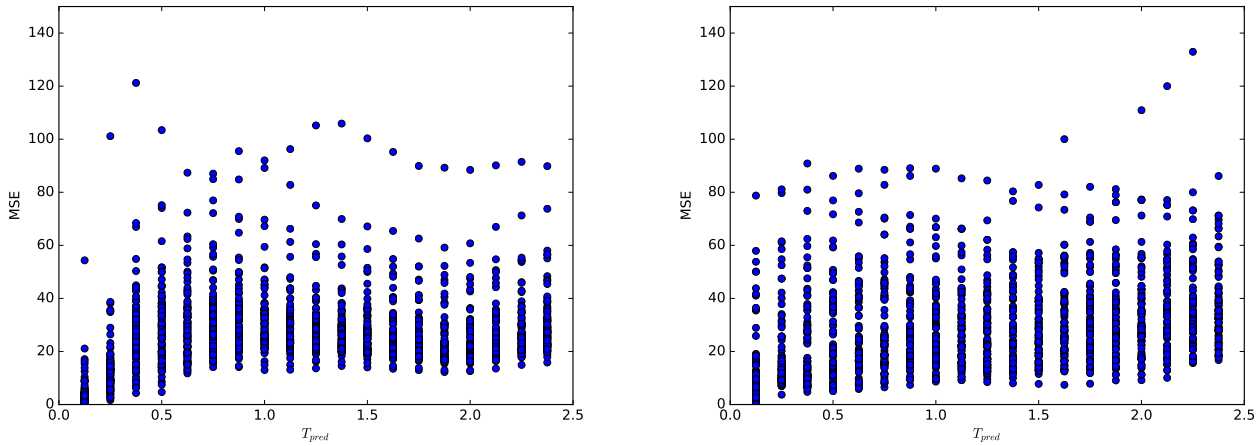


(c) L=5



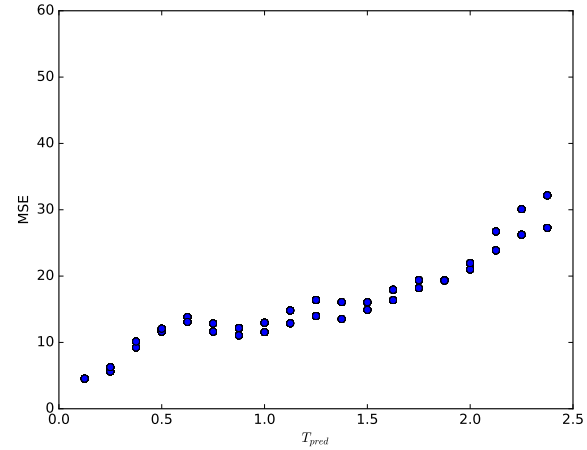
(d) L=7

Figure 2.3: Prediction error as a function of annealing step ($\log \frac{R_f}{R_m}$) for a D=11 Lorenz 96 system. MSE was calculated with $T_{pred} = 2.5$. Structure of the MSE plot on at higher annealing step begins to mimic the structure of the Action plot in 2.2. Thus Action level is a good indicator of prediction capability of an estimate.

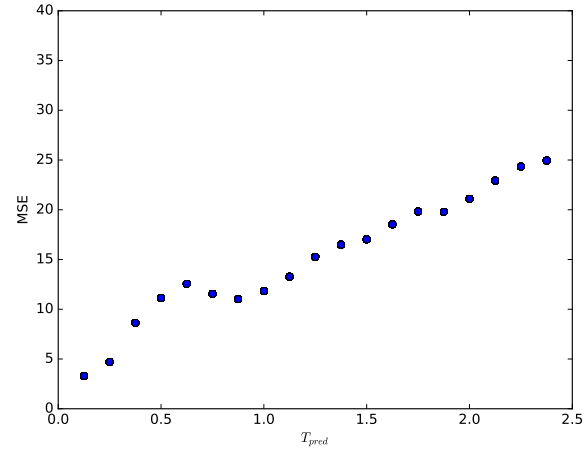


(a) L=2

(b) L=4



(c) L=5



(d) L=7

Figure 2.4: Mean Square Error as a function of Prediction window length for D=11 Lorenz 96 system. Prediction performed at the 40'th annealing step for each L value. MSE increases with increasing T_{pred}. For L= 2, 4 the MSE increases before reaching a maximum value given by the size of the attractor. For L=5, 7 the MSE value has not saturated due to the small initial error.

initial prediction conditions located on the attractor.

This is demonstrated in Figure (2.4). Here the mean square error is plotted as a function of prediction length for each set of initial prediction conditions. The initial prediction conditions were chosen using the 40'th annealing step for each L value. As expected, the value of the mean square error grows with the length of the prediction window, while reaching a plateau for many of the initial conditions on the attractor. For the $L = 5$ and $L = 7$ examples, the MSE does not appear to asymptote by the end of the prediction window. This may simply be due to the small magnitude of Δ at the start of the window.

Figure (2.5) shows the estimation and prediction of the highest likelihood path identified by the lowest prediction value over all annealing iteration and paths. For small L values in which the measured data is insufficient to find an accurate estimate of the states and parameter values [20] the chosen paths are visibly poor at estimating and predicting the trajectory of the plotted state variable. With increasing L the estimation and prediction improves significantly. The trajectories in Figure (2.5) indicate the center of the Monte Carlo search over path space in the $x_0(t)$ direction or equivalently the mean of P_{bias} projected onto the $x_0(t)$ plane in path space.

Random Walk

To estimate the distribution, $P(\mathbf{p})$, a random walk was initialized with 100 walkers starting at points uniformly distributed over an area of path and parameter space with side lengths equal to $1/2$ the bias standard deviation as defined in Section 2.3.1. The random walk was run in Python with a burn in period of length 2,000 steps followed by a 1,000,000 step walk. The location of the walkers in path and parameter space was recorded every 80 steps. This results in 1,250,000 individual paths sampling the state and parameter space.

Figure (2.6) displays the output of the SMC protocol projected into a histogram $P(G)$ in one dimension for the forcing parameter G as described in Section 2.10. The histograms in Figure (2.6) are qualitatively what is expected. The bottom right plot in Figure (2.6) demonstrates

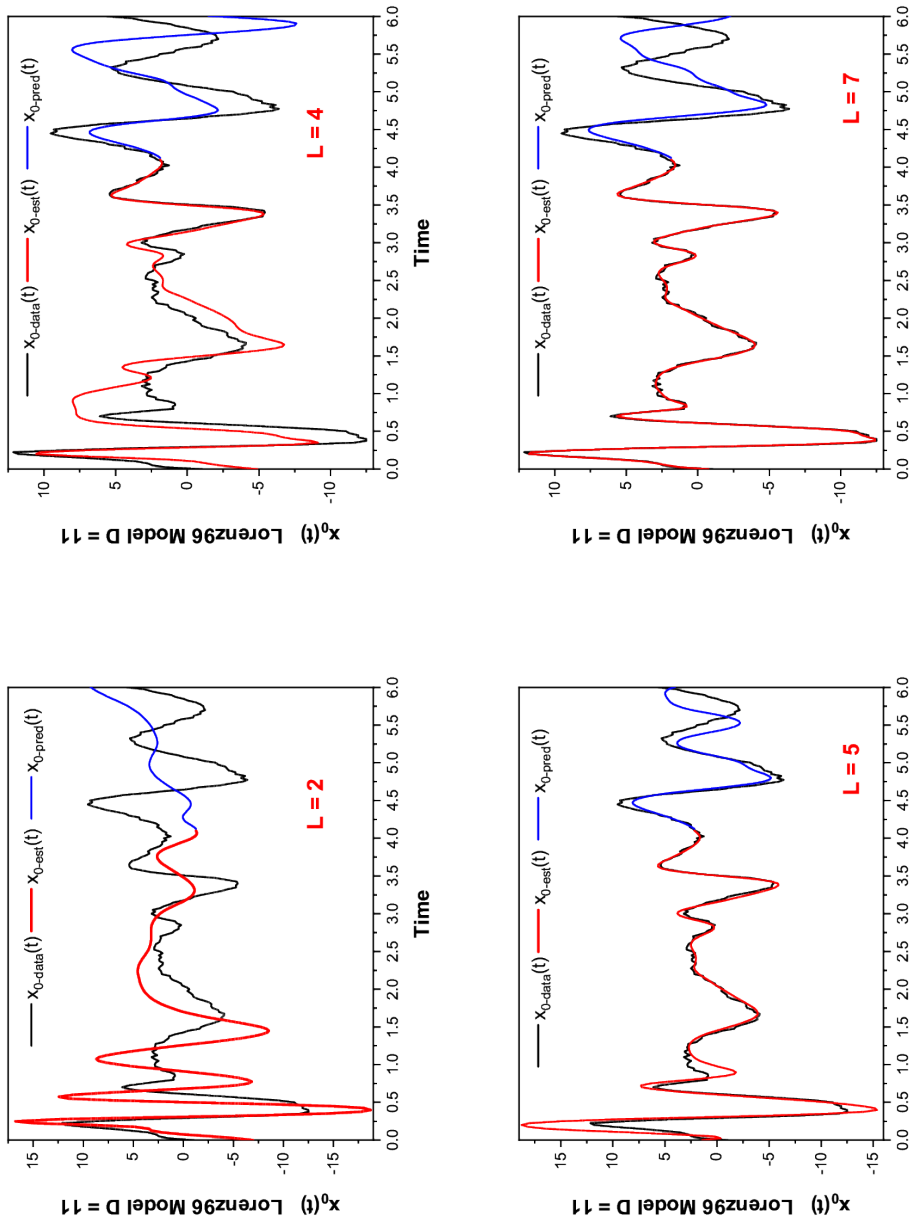


Figure 2.5: $x_0(t)$ data, estimated, and predicted D = 11, L = 2,4,5,7 measured variables as indicated in red in each figure. 100 prediction time points (2.5 time units) past the estimation window. SMC is used to estimate $x_0(t)$ in the observation window and predict beyond that.

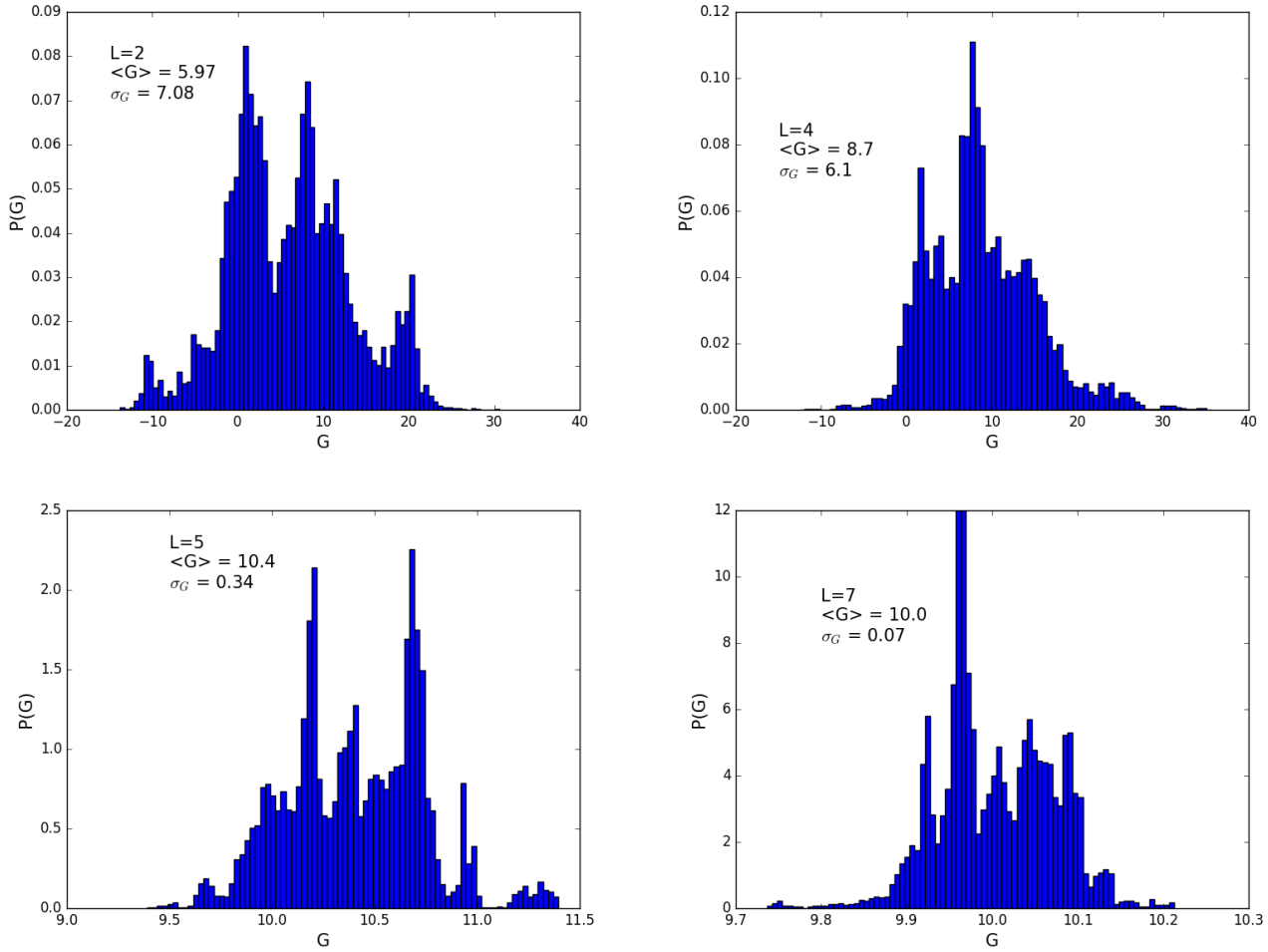


Figure 2.6: Distribution of accepted walker locations as approximation of the underlying probability density function $P(G)$ of the forcing parameter G in the Lorenz '96, D = 11 equations. We show $P(G)$ for L = 2,4,5,7. L is shown in each panel in red. The mean value of the distribution $P(G)$ of the forcing parameter G is shown along with the RMS variation around this mean. As the number of measurements L increases, the distributions have a more and more accurate mean for G and become narrower and narrower.

multiple peaks in probability for 7 measured states with the largest one being much larger than the others. Additionally the width of the distribution tightens as the number of measured states increases. The estimate of the expectation value of the forcing parameter, found in the upper left corner of the histograms, also improves with increased L. This is as it should be as an increase in the number of measured states L implies an increase in the amount of information presented to the calculation. This, in turn, allows for better estimates of unmeasured quantities.

Stability

To test the stability of this method in producing consistent results when estimating the states and parameters of a system the same analysis was performed 80 times for one specific L value. A stability analysis was performed on the Lorenz '96 system with $D = 11$ and $L = 5$. The relatively low L value was chosen so that the distribution being sampled has more than one local maximum. The expectation value of the forcing parameter, G, at every analysis is recorded and ploted in Figure (2.7).

The resultant mean of means for the estimated forcing parameter is 10.26 while the variance of this value is 0.01. The calculated variance of each individual distribution is approximately 0.1. For an estimate produced from a sufficiently well sampled analysis, we expect the variance of the set of expectation values from all analyses to be significantly smaller than the variance of the distribution sampled as the number of analyses increases. At 80 analyses of 1,250,000 points each, the variance of the means is a factor of 10 smaller than the estimated variance of the distribution.

We expect that the stability of the SMC analysis, determined by the variance of the estimated means through several identical analyses, will decrease with the increasing random walk length within SMC. As the length of the random walk increase the estimated distribution converges to the sampled distribution [31]. When samples must be weighted as in the case of Importance sampling, the number of effective samples is smaller than the number of total samples

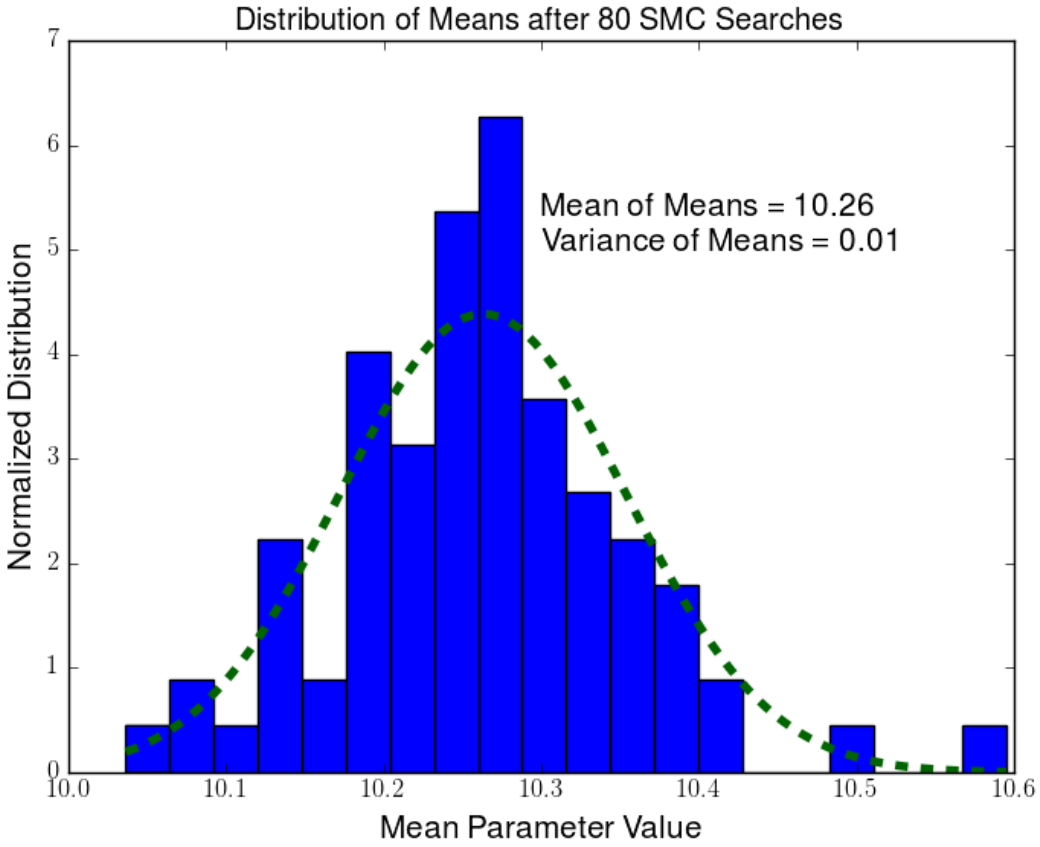


Figure 2.7: Histogram of expectation values of 80 SMC analyses for Lorenz '96 with $D = 11$ and $L = 5$. The mean of the expectation values is 10.26 with variance 0.01. The green dashed line is a Gaussian curve with the same mean and variance as the data points fit to the results.

by a factor depending on the size of the weights [29], in turn determined by P_{bias}. Additionally, the rate of convergence for the estimated and sampled distributions decreases as the dimensionality of the search increases. For these reasons we see a relatively wide distribution for the estimated mean parameter value.

Improved stability may be achieved through increasing the number of samples collected throughout the analysis by increasing the number of walkers, the amount of steps SMC runs for, or both. Increasing the number of samples within the analysis should improve the sampling of the distribution and simultaneously improve the stability of this analysis. In addition to increasing the number of total samples collected, alternative sampling routines which have shorter convergence

times may be implemented to improve the stability of the results [32].

2.5 Discussion

We have shown that using random walk methods as described here to sample conditional probability distributions in the neighborhood of their maximum, informed by VA numerical optimization, provides useful information as to the structure of the probability distribution. Understanding this structure gives us information as to accuracy of parameter and unmeasured state estimates which can be useful in applications where a confidence interval is desired. Additionally, the exploration of the structure of conditional probability densities confirms previous assumptions of the dependence of the probability distribution on the number of measured variables. The SMC method presented here demonstrates the expected smoothing of the probability density function given more information and demonstrates promise in approximating higher order estimates of moments of the distribution given the multimodal structure of $P(\mathbf{X}, \mathbf{p})$.

The analysis of stability of this method in the form of estimated mean and standard deviation of the $P(G)$ distribution demonstrates acceptable stability for $L = 5$ measured variables in a chaotic system. Further analysis must be done to calculate the variation in the estimated mean and standard deviation as the amount of measured data varies. Due to the fact that rate of convergence depends on the level of tuning of step sizes for a particular search distribution and that the optimal step size for a distribution depends on the structure of the search distribution [13], it can be expected that the rate of convergence for the estimate of the probability distribution will vary with varying L values. This may be verified through estimating the variance in the mean SMC parameter estimate as a function of L value and may be addressed through adaptive tuning of step sizes to the distribution, exploring the space for a sufficient amount of time, or choosing search methods which converge to the search distribution at higher rates. Regardless of the exact search method used, SMC demonstrates an informed method through which the search space

of a high dimensional and multimodal probability density may be constrained to only regions of significance. This bounded search decreases the necessary computational time to understand and estimate the regions of the probability density function that dominate the integrals used to calculate expectation values.

This chapter was adapted from work being prepared for publication of the material as it may appear in S. Shirman, H. D. I. Abarbanel, Strategic Monte Carlo Methods for State and Parameter Estimation in High Dimensional Nonlinear Problems, with consent of the authors. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Twin Experiments on Neural Models

3.1 Introduction to Biological Neuron Models

Neurons are one of the building blocks of animal brains and nervous systems. These cells communicate in a network through electrical and chemical signals. The electrical currents in neurons occur across their membrane and travel from their starting point down the extended processes of the cell. Unlike in engineered circuits in which the current is caused by electron flow, the currents in neurons are comprised of the movement of charged ions. The electrical properties of neurons are determined by their type and the time varying environment that they reside in. This thesis focuses on spiking neurons, these are neurons which produce large amplitude and high frequency action potentials which occur as the neuron membrane potential, defined as the potential difference between the inside and outside of the cell membrane, reaches some threshold value. Additionally, all models discussed in this thesis are biophysically realistic conductance based models which describe the membrane voltage evolution as a function of ion flow and chemical and biological states of the neuron.

Conductance based neuron models are of the form

$$\frac{dV(t)}{dt} = C^{-1}\sum_i I_i(t) \tag{3.1}$$

$$I_i(t) = g_i(t)(E_i - V(t)) \tag{3.2}$$

The subscript i refers to the various ions whose currents drive the cell's voltage evolution. For each ion type, the conductance $g_i(t)$ has its own form due to the biological and chemical properties of the cell and its components. The number and type of ion currents present in the cell depend on the cell type and can be determined experimentally through pharmacological means.

3.2 The NaKL Model

The simplest action potential generating neuron model was first described by Hodgkin and Huxley [14]. This model contains two ion currents, Sodium and Potassium, and one "leak" current which corrects for the permeability of the neuron membrane. Due to the currents present within this model, it is called the NaKL neuron model and is listed below.

$$\frac{dV(t)}{dt} = g'_{Na}\cdot m^3(t)h(t)(E_{Na} - V(t)) + g'_K\cdot n^4(t)(E_K - V(t)) + g'_l(E_l - V(t)) + C_m^{-1}I_{inj} \tag{3.3}$$

$$\frac{dx(t)}{dt} = \frac{x_\infty(V) - x(t)}{\tau_x(V)} \tag{3.4}$$

$$x_\infty(V) = \frac{1}{2}\left[1 + \tanh\left(\frac{V-\theta_x}{\sigma_x}\right)\right]; \qquad \tau_x(V) = t_{x,1} + t_{x,2}\left[1 + \tanh\left(\frac{V-\theta_x}{\sigma_x}\right)\right]^2 \tag{3.5}$$

$$x = m, h, n$$

Where the g_x are the maximum conductances of various current types and $g'_x = \frac{g_x}{C_m}$ is the

conductance scaled by the membrane capacitance. The E's are reversal potentials for the various current types and $m(t)$, $h(t)$, and $n(t)$ are gating variables that determine the time dependence of the conductance of the cell to a specific ion type and have a dynamical range between $[0,1]$. The currents are regulated by ion channels that allow ion flow across the membrane. When the channels are 'open' ions can flow, while when the channels are 'closed' they can not. The state of ion channels is determined by the cell environment they reside in. The maximal conductance of a channel is given by the properties of the ion channels themselves and the density of these channels within the cell membrane. The actual conductance of an ion channel is given by the product of the maximum conductance and the probability of an ion channel being open.

The equivalent circuit diagram for this set of equations is given in Figure (3.1) to help visualize the ion flow.

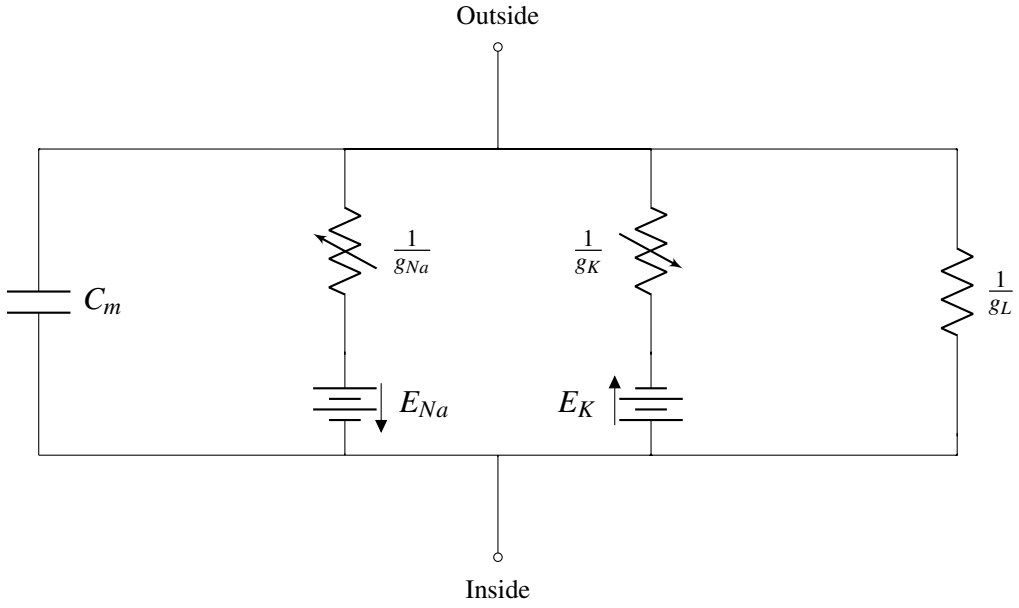


Figure 3.1: Equivalent electrical circuit of a NaKL neuron. Note the change in direction of the Sodium and Potassium 'batteries'. The membrane voltage is measured as the difference $V(t) = V_{inside}(t) - V_{outside}(t)$.

Biologically the concentration of the Sodium ions is significantly higher on the outside while Potassium ion concentration is significantly higher on the inside of the cell membrane

during equilibrium [17]. The Sodium current is activated when the probability of a channel being open, $m^3(t)h(t)$, is large. This probability is the product of the probability of Sodium channels opening, $m^3(t)$, and the probability of Sodium channels closing $h(t)$. The product of two gating variables in the Sodium conductance value causes the channels to open and then rapidly close. While the Sodium channels are open, positively charged Sodium ions move from the outside of the cell to the inside. Because the Sodium channels close based on the behavior of $h(t)$, the Sodium current is a fast and short lived current. On the other hand, the Potassium current is activated by the probability for Potassium channels to open, $n^4(t)$. There is no closing probability allowing the Potassium channels to stay open for a much longer period of time than Sodium channels. While Potassium channels are open, Potassium ions diffuse accross the membrane to the outside of the cell.

During an action potential, or spike, the membrane voltage grows from equilibrium. The Sodium channels open, Sodium ions move into the cell and raising the value of the membrane potential. The Potassium ions then move across the cell membrane in the opposite direction. The speed of the Sodium ions opening, given by the inverse of $\tau_m(V)$, is much larger than the speed at which the Potassium channels open. Thus, overall, the membrane potential rises. As the membrane potential rises, the Sodium channels begin to close due to the effects of $h(t)$, the closing probability. This shuts off the effects of the Sodium current and allows the slower Potassium current to decrease the value of the membrane potential until equilibrium is once again achieved, ending the action potential.

The difference in equilibrium concentrations on the inside and outside of the cell between the two ions leads to very different reversal potentials for these two ion channels. These different reversal potentials effect the direction of the current due to the different ion types. The reversal potential for Potassium is set to $-77mV$ which is approximately equal to the equilibrium potential of the cell while the reversal potential for Sodium is set to $55mV$. These reversal potential differences cause the Potassium current to decrease the membrane potential value for

any membrane potential above equilibrium while the Sodium current increases the membrane potential for membrane potentials below $55mV$. Because the conductance value for the Sodium ions is larger than for Potassium and because the time constant for the opening of the Sodium ion channel is shorter than for the Potassium ion channel, the interplay of Sodium and Potassium currents cause a sharp increase followed by a slower decay in the membrane potential when a positive driving current is present.

We assume in this work that the properties of ion channels are known and constant over cells and that cell to cell variation in conductance properties corresponds to a varying concentration of channels embedded in the membrane leading to variable maximal conductances [3]. For this reason we will set the channel parameters as constant throughout this thesis and will treat only the various maximal conductances and the capacitance of the membrane as unknown.

Results on parameter estimation from simulated experiments will be presented in this chapter. The data used in the experiment will be the voltage time series of a simulated neuron and the aim of the estimation will be to determine the time series of the gating variables and the maximal conductance values using the optimization and random search methods developed in previous chapters. Simulated experiments will be conducted on two different types of conductance-based neuron models with the goal of estimating the value of the maximal conductances and the error bounds or sensitivity of the model to particular estimates.

The NaKL model is employed as a test bed for the Optimization and SMC routine on physically interesting systems. We simulate the time evolution of this neuron using parameters set to accepted values [17]. Normally distributed additive noise is added to the recorded time series of the data. Below is the time series of voltage data and driving current used in this experiment. Only the voltage data is presented to the optimization or SMC procedures for parameter and state estimation.

During the model fit and parameter estimation process all parameters besides the conductances and the membrane capacitance are set to the approximate biological values. See Table

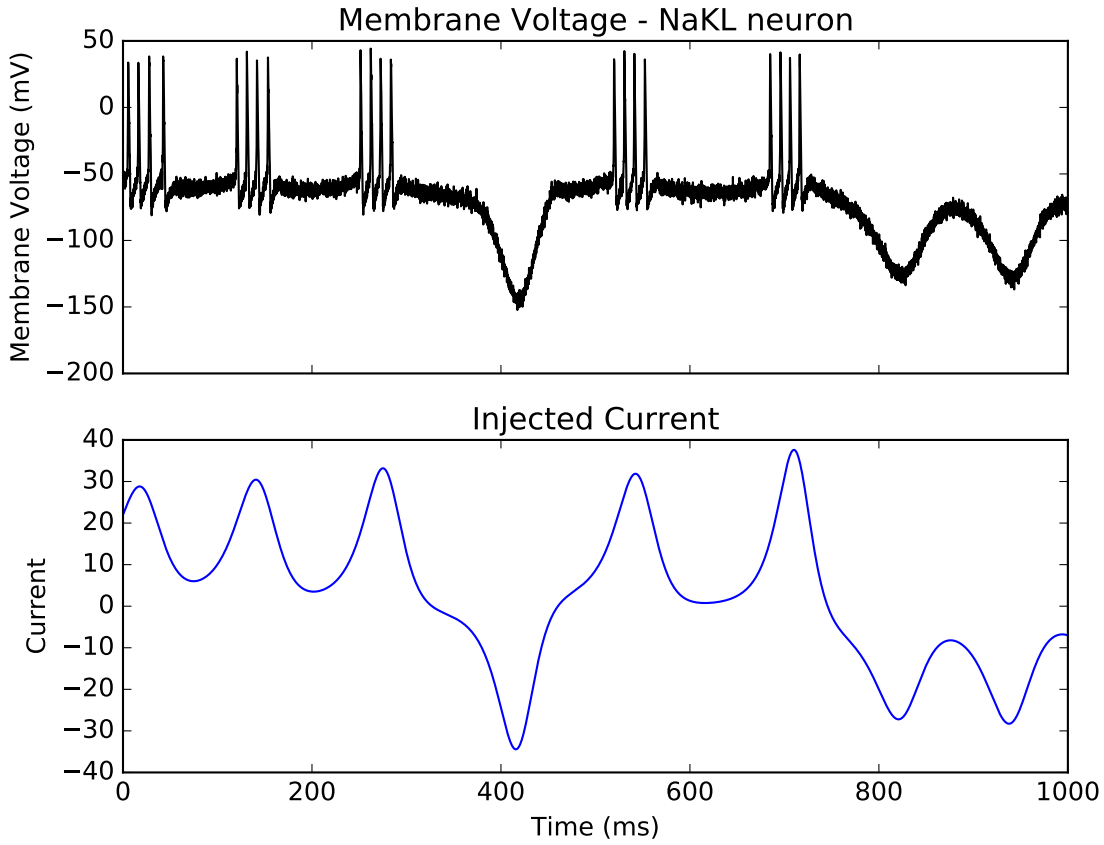


Figure 3.2: Voltage and current time series from the simulated neuron. Only the first 100ms are used for the estimation procedure. The current is measured in μA.

(3.1) for parameter values used in the data simulation and the search range used in the parameter estimation for conductances and membrane capacitance. The estimation of parameters and the three unmeasured states is conducted using data from 1001 current and voltage time points from $0ms$ to $100.1ms$. See Figure (3.2) for data.

3.2.1 Applying Optimization to NaKL Parameter Estimation

For the analysis and parameter estimation from the NaKL simulated data, time points $[0ms, 100.1ms]$ were used. The variational annealing method was applied beginning in such a way that the model precision was increased by a factor of 1.5 at every iteration. In other words,

Table 3.1: Parameter values used for data simulation and model fit. Chosen values are within biologically realistic ranges and fit commonly accepted values. **Left hand table** contains fixed ion channel parameter values used for data simulation and model fit. **Right hand table** contains cell dependent parameter values (maximal conductances and capacitance) used for data simulation and lower and upper bounds for the optimization procedure used for model fit.

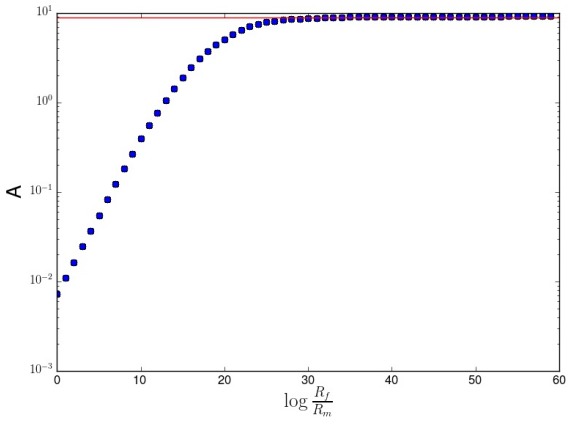
Parameter	Value
$E_{Na}\ (mV)$	50
$E_K\ (mV)$	-77
$E_L\ (mV)$	-54
$\theta_m\ (mV)$	-40
$\sigma_m^{-1}\ (mV^{-1})$	0.0667
$t_{m,1}\ (ms)$	0.1
$t_{m,2}\ (ms)$	0.4
$\theta_h\ (mV)$	-60
$\sigma_h^{-1}\ (mV^{-1})$	0.0667
$t_{h,1}\ (ms)$	1
$t_{h,2}\ (ms)$	7
$\theta_n\ (mV)$	-55
$\sigma_n^{-1}\ (mV^{-1})$	0.03333
$t_{n,1}\ (ms)$	1
$t_{n,2}\ (ms)$	5

Parameter	Value	Lower bound	Upper bound
$g'_{Na}\ (mS/\mu F)$	120	0	200
$g'_K\ (mS/\mu F)$	20	0	100
$g'_L\ (mS/\mu F)$	0.3	0	10
$C^{-1}\ (1/\mu F)$	0.8	0	10

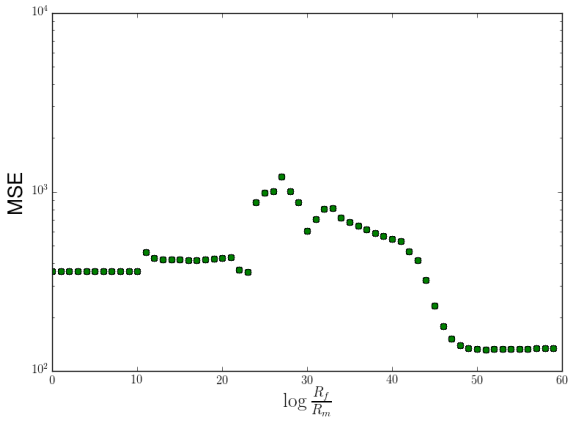
$R_f = R_{f,0}1.5^{\beta}$ where β is the iteration number. The only data presented to the optimization routine was the voltage time series data. Due to the high level of coupling in Equations (3.3)-(3.4), the information present in the voltage time series is sufficient to generate a smooth enough action surface that only one action minimum or probability maximum is discovered.

As in the case of $L = 7$, $D = 11$ of the Lorenz '96 model when we see a single action level appear during the annealing process, there is only one unique prediction level as well. If we were to attempt to estimate and plot the probability distribution given by $P(\mathbf{X}, \mathbf{p}) = e^{-A(\mathbf{X},\mathbf{p})}$ we would expect to see a similar single dominant narrow peak as in the $L = 7$, $D = 11$ Lorenz '96 example.

We choose the path with the lowest prediction error to analyze. Figure (3.4) compares the data to the estimation determined through VA as well as the prediction from this estimation. Though the prediction appears visually very near to the time evolution of the data, the MSE of this path is on the order of 100.



(a) Action level plot. The red line represents the expected value of the Action for the path in which the model error term approaches zero. The path discovered by VA asymptotes to this value.



(b) Normalized prediction error plot. Note the large value of the prediction error. Measurement noise is additive normally distributed $3mV$ noise in this experiment, while the prediction error is much higher due to the structure of the time evolution of voltage.

Figure 3.3: Action and Mean Square Prediction Error as a function of annealing step. In this system, only one level exists in each plot, implying a single probability maximum.

Unlike in the Lorenz '96 model, this dynamical system is not chaotic and therefore we expect to see that the prediction *MSE* does not grow significantly with the length of prediction as it did in the Lorenz '96 case. What we see in Figure (3.5) is jumps in the prediction error as a voltage spikes occur. These jumps in error lead to the large MSE found in Figure (3.3). This is due to the fact that voltage spikes are high frequency and high amplitude events. Small errors in initial conditions may generate timing errors in the voltage spikes that in turn generate very large errors. When these errors are entirely timing errors, you see a negative error followed directly by a positive one for a delayed spike or a positive error followed by a negative one for a preemptive spike. Since the *MSE* is calculated by squaring the error at each time point, a timing error in the spike may cause very large *MSE* values even if the time series visually appears to match the data. Due to this property of the voltage time series, Mean Square Error is likely a poor metric for error in this system. To compensate for this, the lowest error paths chosen as the center of the random search described in the next section were visually inspected to determine plausibility.

To plot the probability distribution and approximate the expectation values of the parame-

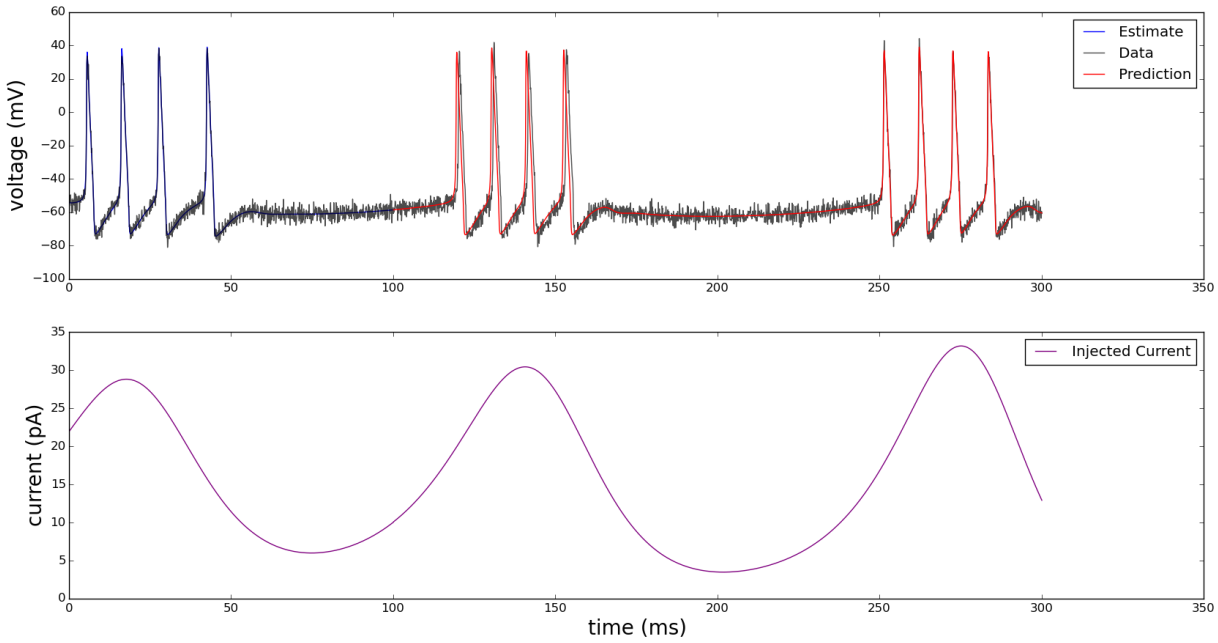


Figure 3.4: In the top plot we display a comparison between the voltage data and the voltage estimation and prediction in turn. Data is in black, estimation is in blue and the prediction is in red. The lower plot contains the driving current for the voltage.

ters of motion and any other path variables, we apply the SMC method developed in the previous chapter.

3.2.2 Applying Strategic Monte Carlo to NaKL Parameter Estimation

Following the optimization procedure we search in regions of high likelihood to determine parameter estimates and the model sensitivity to these parameters. The center of the search and of P_{bias} is chosen as the path and parameter estimates leading to the smallest MSE in combination with visual inspection of the predictions. The path providing the worst prediction is chosen similarly and determines the width of the search constraint. The search is completed with 100 walkers, 1000 burn-in steps, 500,000 search steps, and every 40 accepted paths were saved for analysis. Due to the non-chaotic nature of the system we have chosen to use fewer walker steps

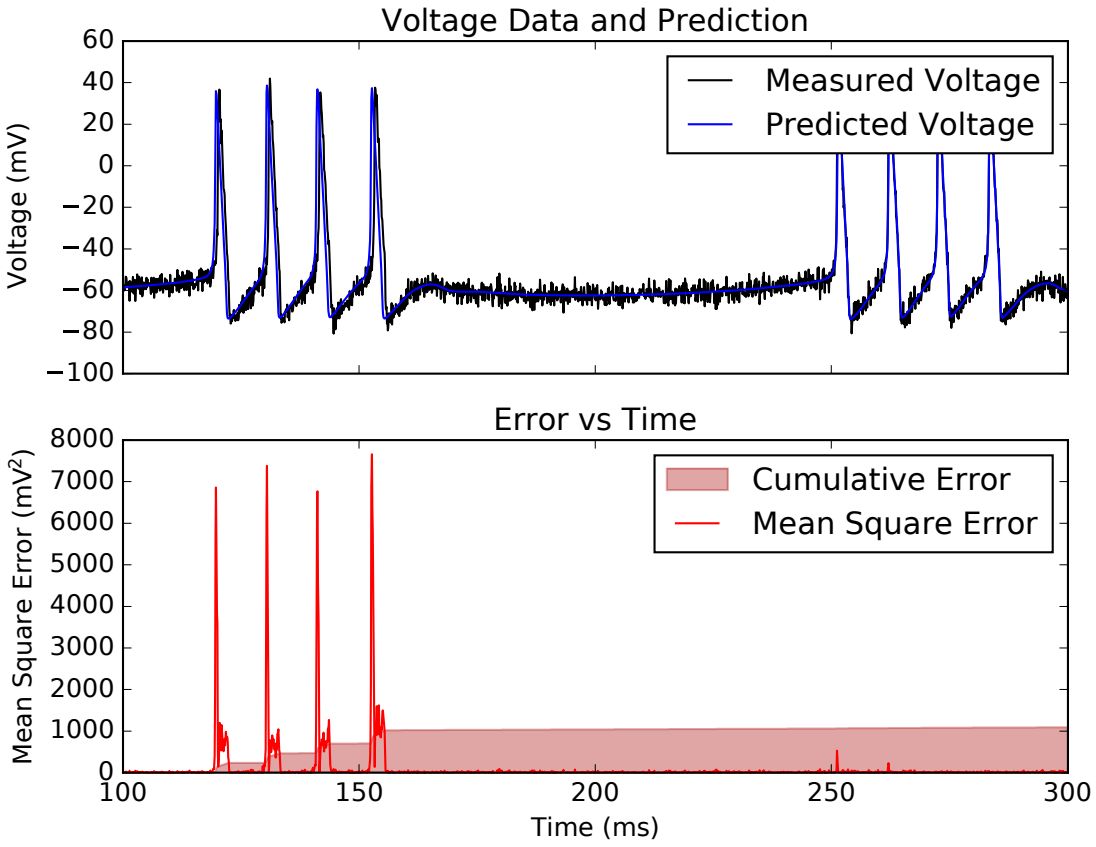


Figure 3.5: The top plot shows a comparison between the predicted path arising from estimates chosen from VA and the data. The lower plot demonstrates the contribution of each time point to the prediction error in the red line while the cumulative error (scaled by a factor of 5 for ease of visibility) is displayed in the shaded region.

than in the Lorenz '96 example as we expect a smoother probability distribution.

The results of the search are surprising and as follows. When the probability distribution that results from SMC analysis of the data is projected down into $1-$dimensional space for each individual parameter, we see approximately two distinct peaks separated by nearly zero probability regions. The expectation values and standard deviation of leak conductance and capacitance of the cell, g_L' and C_m^{-1}, overlap with the true values of these parameters used for the simulation of the data. The ion conductance estimations, on the other hand, do not overlap with their true values.

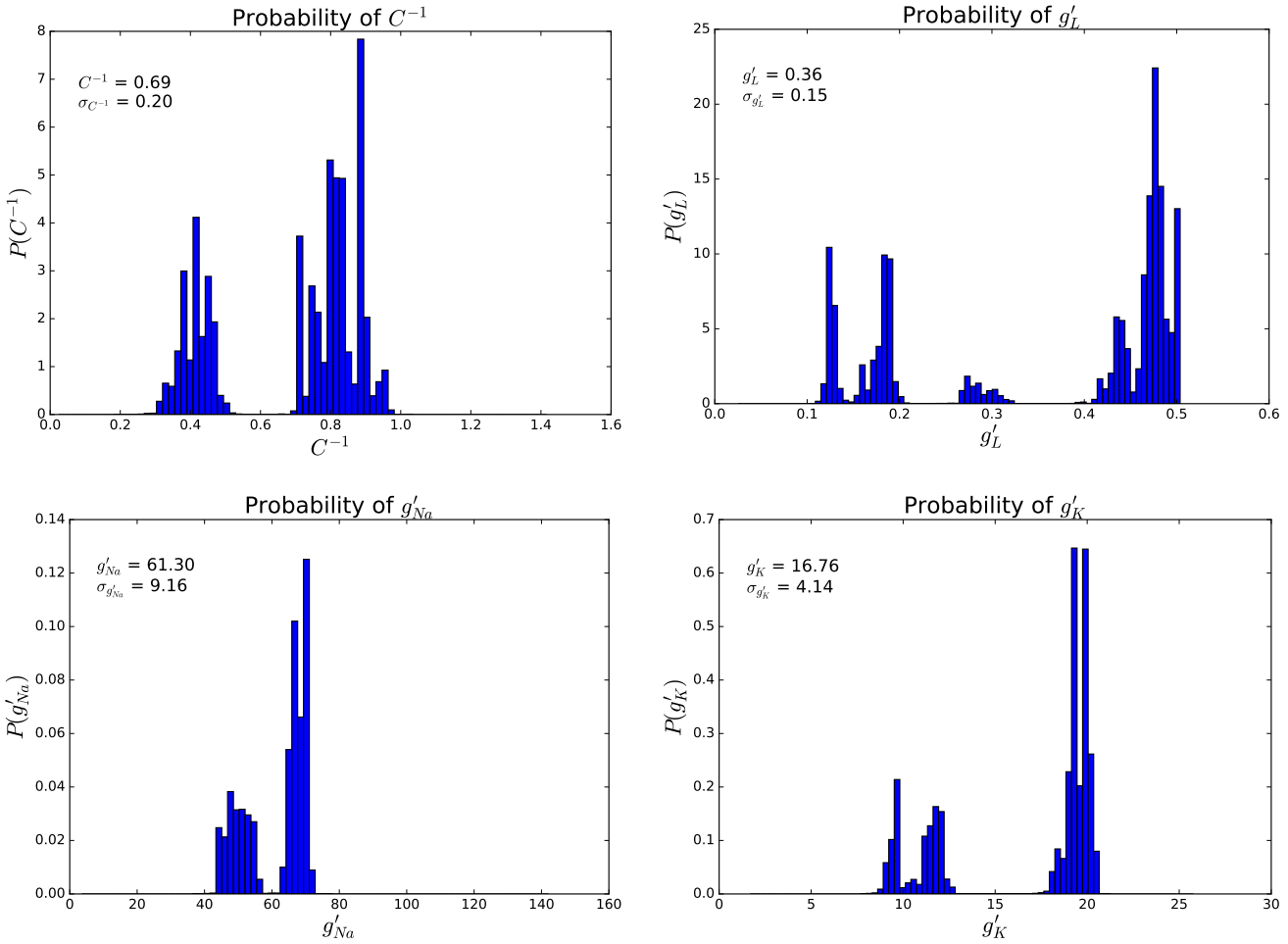


Figure 3.6: 1−dimensional projection of NaKL probability distribution determined by SMC onto paramter axes. Each parameter plot contains two distinct regions of high likelihood.

The structure of the projected distributions in Figure (3.6) look very much like a sum of delta functions as there are very low probability regions separating the rapidly decaying peaks. Due to this structure, it might be more plausible to treat the probability distribution as one of a superposition of two likely states in which each distinct state satisfies the equations of motion but the mixed state does not. In other words, the resultant probability distribution plots imply that Laplace's approximation, in which the most likely state is taken to be the estimated value, is the appropriate approach. In this case we would examine the local probability maxima separately instead of calculating the expectation value.

Since we cannot display the probability distribution function in the $4-$dimensional space of all the parameters we searched, we look at projections of the probability onto the g'_{Na}, g'_K-plane. This particular choice was made after an examination of the cross-correlations of estimated parameter values. The estimation for the two ion conductances were highly correlated, so theis particular plot would be more information rich than any other $2-$dimensional projection. Additionally, these two particular parameter values are related to more interesting terms in the equations of motion. See Figure (3.7) for the $2-$dimensional projection of the probability distribution. As can be infered from the $1-$dimensional projections in Figure (3.6), the $2-$dimensional projection contains two local probability maxima that rapidly decay.

Several identical SMC experiments were run on the NaKL data to determine the stability of the procedure. In each incident, the location of the local maxima were the same but their relative heights changed between experiments. Though one would expect the higher probability peak to produce better estimates and predictions, in Figure (3.7) the lower amplitude peak produced better estimation of the voltage trajectory during the analysis window. The true relative values of the peak amplitudes may be poorly estimated due to the sharpness of the peak relative to the flatness of the probability distribution in the rest of parameter space. To better estimate the relative heights of these maxima future work may implement adaptive random walk methods which are better suited to surfaces in which both large and small gradients exist. Such methods should provide a

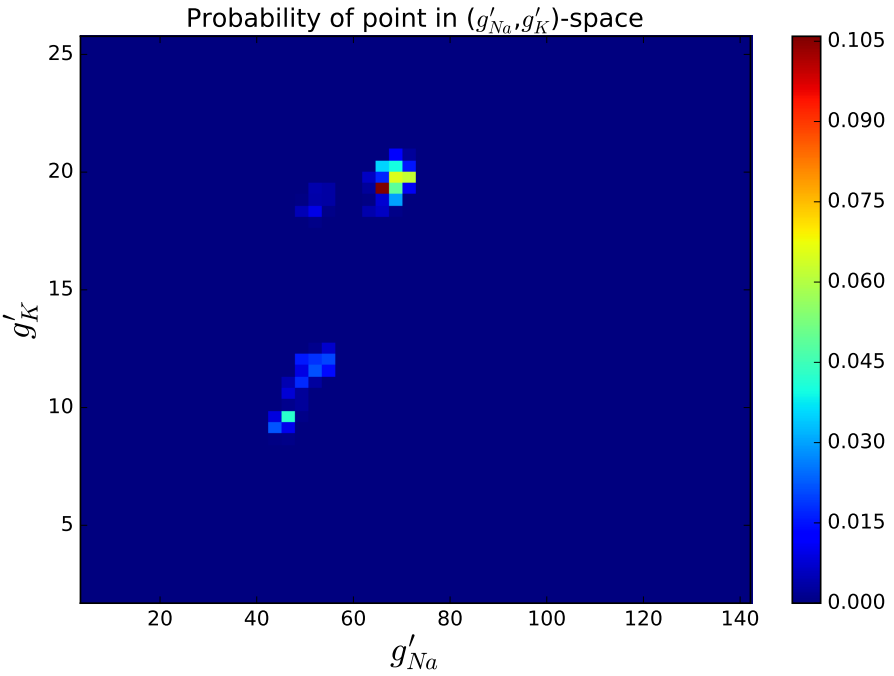


Figure 3.7: $2-$dimensional projection of NaKL probability distribution determined by SMC onto g'_{Na}, g'_K-plane. Local maximum is chosen for parameter estimation due to the rapid decay of the distribution. Width of the local maximum is used to approximate the error in the estimation.

more stable estimate of the structure of the probability maxima.

Though the rapidly decaying local maxima found by SMC do not overlap with the true values for the parameters, we can show that these estimates are a similarly viable set of parameters for the data used in the analysis. If the parameter estimate values in Table (3.2), chosen to be the location of a local maximum of probability, are used to simulate the neuron behavior, using the initial state conditions at the beginning of the estimation window chosen by VA, the estimated and predicted time evolution of the measured state variable is nearly identical to the measured time evolution. The plot in Figure (3.8) demonstrates the degeneracy in the dynamics between the true values and those found by SMC.

It is unclear why the true or correct parameter values are not found through the random walk nor why the degenerate parameter values are not found by VA, but it is interesting that a

Table 3.2: NaKL parameter estimates as found by SMC. These correspond to the lower peak in the distribution shown in Figure (3.7) and were chosen due to their better quality fit compared to the other peak.

Parameter	True Value	SMC Estimate
C^{-1} $(\frac{1}{\mu F})$	0.8	0.43
g'_L $(\frac{mS}{\mu F})$	0.3	0.17
g'_{Na} $(\frac{mS}{\mu F})$	120	47
g'_K $(\frac{mS}{\mu F})$	20	8

degenerate solution that produces equivalent behavior in the estimation regime was found. Future work may focus on understanding why the true values of the parameters (those used to generate the data) were not found through SMC while a different, degenerate set of parameters was.

3.3 Discussion

There are several interesting conclusions that can be made from the analysis in this chapter related to both the definition of the Action and properties related to the system itself. These conclusions may also guide the direction of future work as they open up new questions.

Figure (3.5) demonstrates that the mean square error may be a poor choice for an error metric due to the fact that its largest contributions are the spike events while low frequeny regions influence the total MSE value minimally. The mean square structure was chosen because it is identical to the measurement error term in the definition of the Action in Eq. (1.19). This implies that the chosen structure for the Action and thus the probability distribution may be non-ideal. As the Gaussian structure chosen for the model and measurement terms was chosen arbitrarily for simplicity of calculation, it may be prudent to explore the effect of using other approximations for the delta function in future work on this analysis.

This choice in Gaussian structure for the Action may also contribute to the inconsistency of the relative heights of the two probability maxima determined through multiple attempts of SMC and the inability of SMC to locate the probability maximum found through VA corresponding

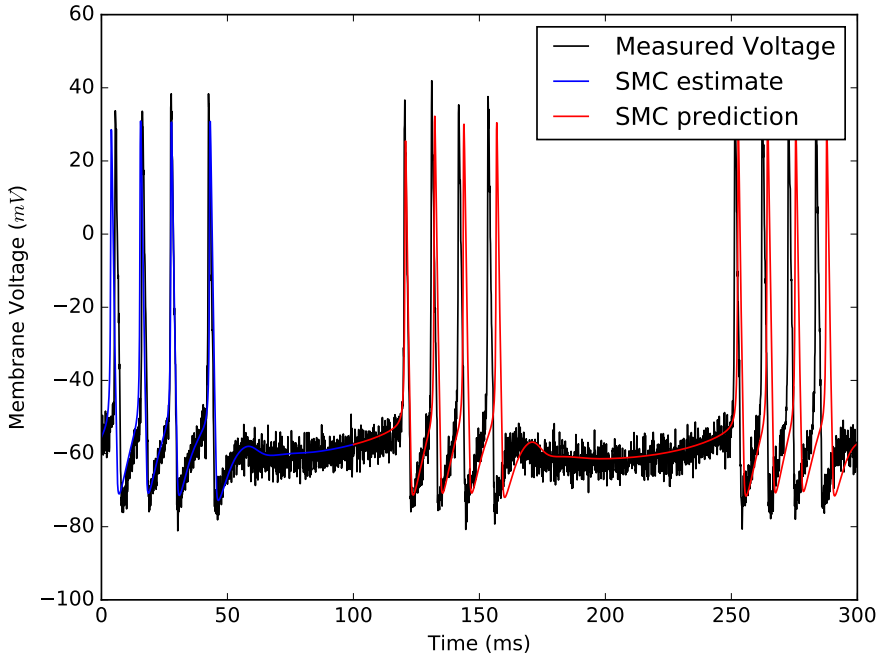


Figure 3.8: Comparison plot of simulated Voltage data and simulation using SMC resultant paramters. The noisy black curve is the data, the blue curve is the trajectory generated by the set of parameters at the local maximum of the probability distribution as determined by SMC and compared to the voltage trajectory within the estimation window, the red curve is the trajectory generated by the same SMC parameters and compared to a portion of the time trajectory not used for estimation.

to the 'true' parameter values. Because of the Gaussian structure of the Action the resultant surface shown in Figure (3.7) contains two very narrow and sharp maxima separated by flat regions. Such surfaces are difficult to explore thoroughly with a random walk Monte Carlo as there is no gradient information in the flat regions to guide the walk toward probability maxima. Additionally the ideal step size for the flat regions is significantly different from the step size which would most efficiently and accurately search and estimate a narrow and sharp peak. For this reason, adaptive search methods in which the local gradient dictates the size step size of the search in that region may be best for neuron models. As the Gaussian structure was chosen arbitrarily and may be a poor choice in approximating the error, as previously discussed, SMC analysis using different Action strucutres may improve the likelihood of finding a local probability

maximum that corresponds to the set of parameters used to simulate the data. Though adjusting the search method and the definition of the Action may provide accurate results, the preliminary ones presented in Figure (3.7) validate assumptions made in previous optimization work on neuron models: that the probability distribution contains sharp and narrow local maxima and that the Laplace approximation is appropriate and valid when estimating the values of model parameters.

Since a different and degenerate set of parameters were discovered through SMC that were not found using VA, future work should involve determining which methods can be used to distinguish between two local probability maxima, particularly in the case of the distribution being a superposition of several distinct delta functions. One possible avenue of research would be to study the effects of varying the driving current on the structure of the probability distribution. As the driving current determines where in state space around the attractor the neuron resides, the choice of this driving current may be important in providing enough information to determine the shape of the attractor and the model parameters that generate the measured trajectory.

Chapter 4

Machine Learning as an application of Data Assimilation

Machine learning (ML) is a field of study in which computers learn to make various decisions without explicitly coded learning rules. The machine is trained on a set of data with the goal being to learn correlations or properties of that data by adjusting internal parameters to best fit with the data. Once it has 'learned' and its internal parameters are set, it is tested on new data from the same set to determine if it has correctly tuned itself to the data. Machine learning can be applied to many types of problems, but this chapter will focus specifically on machine learning as applied to problems of catagorizing data. One such categorization problem is discerning the appropriate labels for a set of images. For example, you may have a data set of images of pets and you want the machine to learn to label a presented image as "bird", "dog", "cat", or "fish".

There are two main types of machine learning: Supervised and Unsupervised learning [11]. Supervised learning involves presenting the machine with an input and an output data pair during training. For example the data presented to the machine while it learns may consist of several images of each type of pet and the label associated with those images. After training is complete an unlabeled image is presented to the machine and its response is compared to

the correct label determined by a human "examiner". Unsupervised learning does not involve presenting the machine with appropriate labels for images during the training period. This type of learning relies on the machine's ability to determine characteristic features for the images, place each image in feature space, and split the feature space in such a way that the boundaries minimize the volume of a cluster of images in features space. Only supervised learning will be addressed here.

This chapter will focus on the parallels between DA and ML. Methods from DA will be applied to a toy problem to show proof of concept and to a classic ML problem to demonstrate feasibility. Both examples demonstrated in this chapter will be of supervised learning of labels of image data. The machine will be presented with a set of grey scale images in the form of an array of pixel intensities and a set of corresponding labels. For a system with D_L labels, the labels are in the form of a one dimensional array of length D_L. Label arrays consist of $D_L - 1$ zeros and one non-zero value. The non-zero value corresponds to the category of the matching image. For example, if two labels are needed to describe the data, the labels would be $\binom{1}{0}$ and $\binom{0}{1}$. The goal of the learning process of ML is to adjust the machine parameters in such a way that the machine output, given by the transformation of the image input data by the underlying machine equations, matches the desired output, the labels. Before diving into the details of the mathematical comparison between ML and DA, we will present a short description of a current ML method.

4.1 Deep Learning

Deep learning is one method through which machine learning is accomplished. In this method a network of nodes is created that connect the input to the output. Individual nodes accept a numerical input and generate a numerical output. Assempled in the achitecture of the network, the nodes jointly represent a transition function fron input data to the network output, typically

estimated label values.

A common architecture for in deep learning is a layered network of nodes [7]. The first layer, called the input layer, receives the data as its input and thus is the same size as the presented data. The final layer of the network, called the output layer, produces the final result of the network: an estimated label associated with the data presented at the input layer. Between the input and output layers there may be one or more 'hidden' layers, so called because their behavior is not seen by the researcher nor is it directly affected by the data.

Each node's behavior is governed by an 'activation function', or a function that defines the output, or state of the node, given an input. The input of a node is a linear combination of a subset of the other nodes in the network. The structure of the network determines the details of this linear combination. We focus only on 'feed forward' networks in which the input of a node in one layer is a weighted linear combination of the states of the nodes only in the layer directly before it. There is no direct coupling between nodes in the same layer or feedback from a layer to another 'behind' it. Mathematically stated, this is:

$$\mathbf{x}^{(l_F)} = \mathbf{F}(\mathbf{x}^{(0)}) \tag{4.1}$$

$$\mathbf{F}(\mathbf{x}^{(0)}) = \prod_{i=1}^{(l_F)} \mathbf{F}^{(i)} \mathbf{x}^{(0)} \tag{4.2}$$

$$\mathbf{x}^{(i)} = \mathbf{F}^{(i)} \mathbf{x}^{(i-1)} = \mathbf{f}^{(i)}_{activation} \left(\overline{\mathbf{W}}^{(i-1)} \cdot \mathbf{x}^{(i-1)} \right) \tag{4.3}$$

The superscript represents the layer in the network and the length of the $\mathbf{x}^{(i)}$ vectors depends on the size, $l_H(i)$, of the corresponding layer. Superscripts range from 0 to l_F implying $l_F + 1$ total layers in the network. We define $\mathbf{x}^{(l_F)}$ as the output, or state, of the final layer of the machine and $\mathbf{x}^{(0)}$ as the state of the first layer of the machine. The state $\mathbf{x}^{(0)}$ is equivalent to the input to the machine.

The matrix, $\overline{\mathbf{W}}^{(i-1)}$ is a $l_H(i) \times l_H(i-1)$ matrix of correlation strengths between the nodes

of the $i-1$'th and the i'th layers of the network. Within this matrix, the (j,k) element is the weight of the connection from the k'th element of the $i-1$'th layer to the j'th element of the i'th layer. The inner product in Eq. (4.3) is carried out using traditional matrix multiplication.

We define $\mathbf{F}^{(i)}$ as the matrix operator that applies the activation function, $\mathbf{f}^{(i)}_{activation}$ on the weighted sum of the state $i-1$'th layer to generate the state of the i'th layer. The range of $\mathbf{f}^{(i)}_{activation}$ is typically chosen to stay within the range of values $[0,1]$ so that an increasing the number of hidden layers does not produce a corresponding increase in the magnitude of the network output. We define the activation function in such a way that it is applied element-wise to the vector produced by the matrix product between $\overline{\mathbf{W}}^{(i-1)}$ and $\mathbf{x}^{(i-1)}$.

The main premise of deep learning is that any function, $\mathbf{F}$, can be described by a composition of continuous, nonlinear $\mathbf{f}^{(i)}_{activation}$ functions given enough hidden nodes [6, 15]. To this end, deep learning networks are generated with as a large a number of layers as can computationally be achieved. The problem then becomes tuning the values of $\overline{\mathbf{W}}^{(i)}$ and the parameters of $\mathbf{f}^{(i)}_{activation}$ in such a way that the overall function $\mathbf{F}$ fits the data. Once the parameters are properly fitted to the data, the network is fixed and tested on new and previously unused data. The output of the network is compared to known labels to determine the quality of the $\overline{\mathbf{W}}^{(i)}$ estimates.

Traditionally, in the field of machine learning, the parameters of the network are tuned in such a way as to minimize a cost function:

$$C = C\left(\mathbf{y}_{label} - \mathbf{F}(\mathbf{y}_{image})\right) \tag{4.4}$$

Where, consistent with the notation in the rest of this thesis, $\mathbf{y}$ is the measured data and $\mathbf{F}$ is the function defined in Eq. (4.1). The cost function is solely a function of the difference between the label data and the estimated network dynamics applied to the image data. There are many forms of this cost function; a common and familiar one is the Gaussian cost function which takes the form

$$C = \left(\mathbf{y}_{label} - \mathbf{F}(\mathbf{y}_{image})\right)^2 \tag{4.5}$$

In both Eq. (4.4) and Eq. (4.5), the model is strongly constrained. As with the dynamical systems examples in previous chapters, this high dimensional, strongly constrained cost function is very difficult to impossible to minimize. Additionally, it is unlikely that without some prior knowlege of the structure of **F** the arbitrarily chosen network structure and the linear combination of activation functions is sufficiently flexible to describe every necessary transformation, **F**, from image to label. To facilitate the minimization of the cost function defined with the nonlinear constraints of the neural network structure and to account for the errors that will exist due to poor model choice we apply methods of annealing in the generation of weight matrix estimates that best fit the provided data.

4.2 Layer to Time equivalence

4.2.1 Defining the Network

We begin by proposing a network made up of l_F layers in which each layer has the same number of nodes, D. We set the connectivity of the network as all-to-all between a layer and its subsequent layer. The parameter values at each layer may vary, but the structure of the transformation function is conserved from layer to layer. Mathematically this implies that the $\mathbf{f}^{(i)}_{activation}$ in Eq. (4.3) are not explicitly dependent on i

$$\mathbf{x}^{(i)} = \mathbf{F}^{(i)}\mathbf{x}^{(i-1)} = \mathbf{f}_{activation}\left(\sum_j \overline{\mathbf{W}}^{(i-1)} \cdot \mathbf{x}^{(i-1)}\right) \tag{4.6}$$

$$\mathbf{x}(i) = \mathbf{f}_{activation}\left(\sum_j \overline{\mathbf{W}}(i-1) \cdot \mathbf{x}(i-1)\right) \tag{4.7}$$

In Eq. (4.7) we have simply transformed the notation for parameter and state variables from a set of $l_F - 1$ discrete weight matrices and l_F state vectors to a notation in which there is only one layer dependent weight matrix and state vector. Eq. (4.7) is equivalent in form to a difference equation for a dynamical system where i is the 'time' variable and the parameters of motion, $\overline{\mathbf{W}}$, are 'time' dependent.

A generalization that is made in the machine learning framework that does not often exist in the classical framework is that the length of the state vectors $\mathbf{x}$ may be time/layer dependent. In machine learning, the length of the state vector $\mathbf{x}(i)$ denotes the number of nodes in the i'th layer of the network. When the length of $\mathbf{x}(i)$ is not constant, the matrix $\overline{\mathbf{W}}(i)$ is not square. The creation or elimination of "states" as a function of layer in an artificial neural network is not new and may be thought of as analogous to the change in number of states of a dynamical system due to the creation or annihilation of particles. See Figure (4.1) for an example of the structure of such a network with varying layer size.

Because the learning phase of ML consists of finding values of $\overline{\mathbf{W}}(i)$ which satisfy the difference equation given some data on the boundaries, ML is analogous to the two point boundary value problem solved previously in this dissertation [2]. As in the dynamical systems case, the cost function in Eq. (4.5) is difficult to estimate and the model chosen in Eq. (4.7) may be insufficient or incorrect to properly describe the dynamics of the system. For these reasons we relax the constraints and allow

$$\mathbf{x}(i) = \mathbf{f}_{activation}\left(\sum_j \overline{\mathbf{W}}(i-1)\cdot\mathbf{x}(i-1)\right) + \boldsymbol{\eta}(i) \tag{4.8}$$

Here $\boldsymbol{\eta}(i)$ is once again stochastic error in the model equations. We may generalize the Action used in previous chapters, defined in Eq. (1.19) to this situation

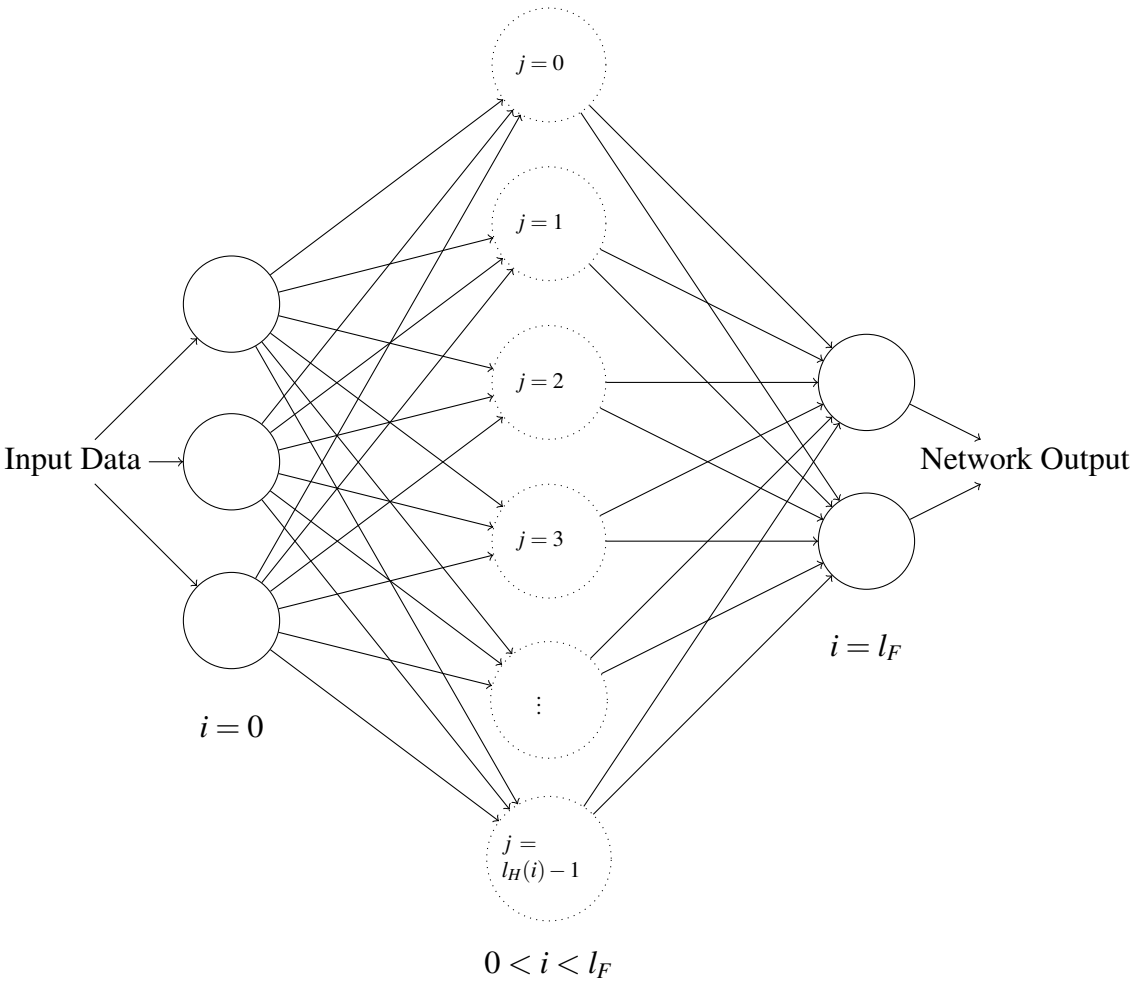


Figure 4.1: Diagram of the feedforward network described in Eq. (4.2.1). Node and layer indices match the notation used to define the weights and the cost function. Middle layer represents $l_F - 2$ hidden layers with the same all-to-all, feedforward connectivity as shown.

$$A(\{\mathbf{x}\},\{\overline{\mathbf{W}}\}) = \sum_i \frac{R_f}{2}\left[\mathbf{x}(i) - \mathbf{f}_{activation}\left(\sum_j \overline{\mathbf{W}}(i-1)\cdot\mathbf{x}(i-1)\right)\right]^2 + \frac{R_m}{2}\left[(\mathbf{x}(0)-\mathbf{y}_{input})^2 + (\mathbf{x}(l_F)-\mathbf{y}_{output})^2\right] \tag{4.9}$$

As previously, the terms squared in Eq. (4.9) are taken to be a vector inner product. In the ML situation, the independent variable, s, is now replaced by the layer number, i. The R_m term is equivalent to the one seen previously in the dynamical systems case for the situation in which data only exists at the boundaries. An example of a dynamical system in which only the boundary

values of the states of interest are measured can be found in the paper by E. Armstrong, et al on flavor evolution of neutrinos [4]. As before, R_f refers to inverse square of the standard deviation of $\boldsymbol{\eta}(i)$, R_m refers to the inverse square of the standard deviation of the measurement noise, and the limit of $R_f \to \infty$ is equivalent to the strong constraint version of this problem defined by Eq. (4.5). Utilizing variational annealing, we begin with an unconstrained optimization such that $R_f = 0$ and slowly enforce the constraints by increasing the value of R_f.

If the network is able to generate the transformation from the input to the output and an appropriate set of weights is discovered, the value of A will plateau to the value of the measurement noise at high R_f. This assertion and the use of the Action in determining the ability of the estiamated weights to generate the necessary transformation is based on the equivalence between the ML problem and the dynamical systems one solved in previous chapters and will be demonstrated later in this chapter.

4.2.2 Action Normalization

Given some set of data we wish to determine a set of network parameters and a network model based on the assymptotic Action value with the increase of R_f. If we analyze a set of data using several varying neural networks the Action defined in Eq. (4.9) must be properly normalized to allow for direct comparison between the performance of different networks.

We assume for a specific classification problem that the input and output layers are fixed due to fixed image and label size. If we wish to relax this constraint within a problem type, the second term in Eq. (4.9) would scale additionally with the size of the input and output layers.

In contrast, within a single classification problem the size of the hidden layers can drastically vary without changing the structure of the data. Due to this, we must take into account the scaling of the Action with the size of the hidden layers. Each element of the first sum in Eq. (4.9) scales with $l_H(i)$, or the size of the layer over which the model is compared to the state estimate. Rescaling the R_f value will not affect the results of optimizing the rescaled Action

through an annealing method because the magnitude of the model error term with respect to the measurement error term varies during variational annealing and because when an ideal minimum is found such that the model perfectly fits the state estimates, the Action becomes independent of the size of R_f . We can rewrite Eq. (4.9) in the following way

$$\begin{aligned} A(\{\mathbf{x}\},\{\overline{\mathbf{W}}\}) = &\sum_i \frac{R'_f(i)}{2l_H(i)} \left[\mathbf{x}(i) - \mathbf{f}_{activation}\left(\sum_j \overline{\mathbf{W}}(i-1)\cdot\mathbf{x}(i-1)\right)\right]^2 \\ &+ \frac{R_m}{2}\left[(\mathbf{x}(0)-\mathbf{y}_{input})^2 + (\mathbf{x}(l_F)-\mathbf{y}_{output})^2\right] \end{aligned} \tag{4.10}$$

Here we define $R'_f(i) \equiv R_f l_H(i)$. It is plain to see that the value of the Action has not changed. We now characterize the method through which R_f can be increased by defining

$$R'_f(i) = R_{f,0}(i)\alpha^{\beta} \tag{4.11}$$

For R_f to increase, α must be a number greater than one and we define β as the iteration in the annealing procedure.

The expression

$$\frac{1}{l_H(i)}\left[\mathbf{x}(i) - \mathbf{f}_{activation}\left(\sum_j \overline{\mathbf{W}}(i-1)\cdot\mathbf{x}(i-1)\right)\right]^2 \tag{4.12}$$

defines the average error per node in the i'th layer. To directly compare two networks with different sized hidden layers the value of $R_{f,0}(i)$ chosen depends on the size of the hidden layers. But because the value of the Action becomes independent of R_f at sufficiently high β value, the annealing procedure may be run on both networks until they have each reached assymptotic Action values. Due to the assymptotic behavior of the Action at high R_f values and the expression in Eq. (4.10) of the Action with respect to the average node error it is now possible to compare the the Action for networks with different layer sizes directly.

The relative values of $R'_{f,0}(i)$ in each layer are arbitrary. Setting the $\frac{R'_{f,0}(i)}{l_H(i)}$ to a constant value for all i will enforce the model on average equally over all nodes because the model error term from a single layer depends only on its average value. Setting $R'_{f,0}(i)$ to a constant will enforce the model in such a way that errors in larger layers are more heavily penalized than errors in smaller layers by a factor given by the layers' relative sizes because the layer's contribution to the model error term of the Action is the total error within that layer.

4.3 Image Recognition

Images can be thought of as comprised of a combination of a set of basis 'features'. These features may be colors, curves, edges, or corners and depend on the set of images being studied. To properly categorize an image it must be transformed into feature space. To allow for a useful mapping the size of the hidden layers must be equal or larger to the dimension of the feature space. For experimental data and non-simulated images the dimension of feature space is impossible to determine prior to testing a network and determining the smallest network able to produce accurate labels subsequent to the learning stage. In Deep Learning the network is structured to have as many layers and as large of layers as computationally possible. This expands the input data into a very high dimensional space within the hidden layers before projecting back down into the dimension of the label. Once the dimension of the hidden layers excedes that of the feature space, additional information cannot be extracted from the images and the accuracy of the network predictions will stabilize. The practice of setting the hidden layer size to the largest feasible value ensures that the minimum dimension of the hidden layers is met, though comes with additional computational costs.

This chapter will explore the quantitative effects of network size on weight matrix estimation and label prediction quality. The Action will be proposed as a possible metric for adaptively tuning the size of the network to optimally balance computational efficiency and accuracy.

4.3.1 Multiple Data Points

Each image is the input of the network while the label is the desired output. The image is presented to the network as a flattened array of pixel values. As each image only contains a fraction of the possible features that the network must be able to distinguish, it is necessary to train the network on an ensemble of images sampled from the database. This ensures that the network has been presented with at least one, but ideally more than one, example of every feature it must be able to map. As the researcher does not know *a priori* what the feature space is, it is common to train on as many images as possible. Typical numbers of images presented to a network in machine learning are on the order of several thousand [26]. Due to computational constraints and the efficiency of Variational Annealing, this chapter will be dealing with smaller numbers.

The network is presented with a large, M-element, set of images with their corresponding labels. The Action defined in Eq. (4.9) allows for only one label-image pair. We want to find a single network, and thus a single set of weights, that fits all label-image pairs within the data set simultaneously. We take advantage of the fact that the Action is the log likelihood and that a joint probability of two independent data points may be determined simply by multiplying them.

$$P\left(\{\mathbf{x}\}_m, \left\{\overline{\mathbf{W}}\right\}\right) \propto e^{-A(\{\mathbf{x}\}_m, \left\{\overline{\mathbf{W}}\right\})} \tag{4.13}$$

$$P\left(\{\{\mathbf{x}\}_0, \ldots, \{\mathbf{x}\}_M\}, \left\{\overline{\mathbf{W}}\right\}\right) = \prod_{m=0}^{M} P\left(\{\mathbf{x}\}_m, \left\{\overline{\mathbf{W}}\right\}\right) \tag{4.14}$$

$$P\left(\{\{\mathbf{x}\}_0, \ldots, \{\mathbf{x}\}_M\}, \left\{\overline{\mathbf{W}}\right\}\right) \propto e^{-\sum_{m=0}^{M} A(\{\mathbf{x}\}_m, \left\{\overline{\mathbf{W}}\right\})} \tag{4.15}$$

$$A\left(\{\{\mathbf{x}\}_0, \ldots, \{\mathbf{x}\}_M\}, \left\{\overline{\mathbf{W}}\right\}\right) = \sum_{m=0}^{M} A(\{\mathbf{x}\}_m, \left\{\overline{\mathbf{W}}\right\}) \tag{4.16}$$

The m subscript in the above equations corresponds to a specific label-image pair. The weights do not have an index over the data pairs because we wish to find a single network that is

capable of transforming the input data into the appropriate label for all elements of the data set. The minimization of the Action in Eq. (4.16) occurs in the space of all $\{\mathbf{x}\}_m$ and $\{\overline{\mathbf{W}}\}$. When the set of weights is chosen such that the Action is minimized and the R_f (model error) term approaches zero, the Action simplifies to the sum of the R_m (measurement error) terms in Eq. (4.9) over all data points. At this point the Action should be equivalent to the expectation value of the noise in the image recording.

To allow for the number of label-image pairs to grow without overwhelming the calculation, we scale the Action by M. The location of the minimum of this new Action shown in Eq. (4.17) will be the same as the one in Eq. (4.16).

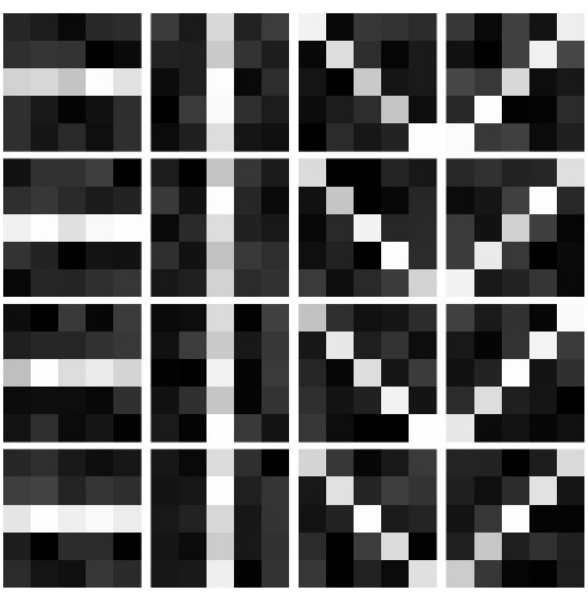
$$A\left(\{\{\mathbf{x}\}_0,\ldots,\{\mathbf{x}\}_M\},\{\overline{\mathbf{W}}\}\right)=\frac{1}{M}\sum_{m=0}^{M}A(\{\mathbf{x}\}_m,\{\overline{\mathbf{W}}\}) \tag{4.17}$$

$$\begin{aligned}A\left(\{\{\mathbf{x}\}_0,\ldots,\{\mathbf{x}\}_M\},\{\overline{\mathbf{W}}\}\right)=\frac{1}{M}\sum_{m=0}^{M}\sum_{i}\frac{R'_f(i)}{2l_H(i)}&\left[\mathbf{x}^m(i)-\mathbf{f}_{activation}\left(\sum_j\overline{\mathbf{W}}(i-1)\cdot\mathbf{x}^m(i-1)\right)\right]^2\\&+\frac{R_m}{2}\left[(\mathbf{x}^m(0)-\mathbf{y}^m_{input})^2+(\mathbf{x}^m(l_F)-\mathbf{y}^m_{output})^2\right]\end{aligned} \tag{4.18}$$

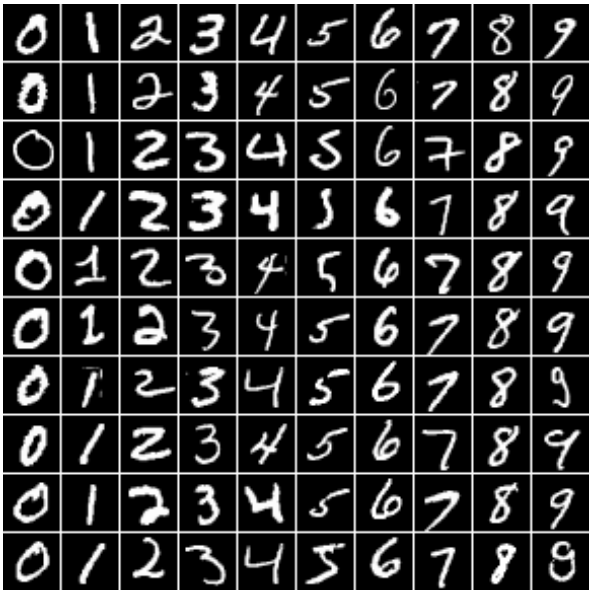
4.3.2 Examples

We present two sents of examples of image recognition using VA methods to minimize the Action while training the network. The first example uses a data set of simulated images of bars. These simulated images are 5x5 pixels in size and the single pixel wide bars are oriented vertically, horizontally, or diagonally. Each bar passes through the center of the image. A 'basis' set of images was generated with the bar at pixel value 1 and the blank space at pixel value 0. The noisy data was generated by adding 17% uniformly distributed uncorrelated noise to the basis images and producing 8000 total examples of the four types of images. Figure (4.2a) contains a representative sample of the images in this data set. Each image is labeled with the appropriate

4 dimensional label vector: $(1,0,0,0)$ for horizontal, $(0,1,0,0)$ for diagonal from top to bottom, $(0,0,1,0)$ for vertical, and $(0,0,0,1)$ for diagonal from bottom to top. The second example uses a well known and common test set of handwriting samples, MNIST [25]. This data set contains 28x28 pixel images of handwritten digits collected from both high school students and government employees. As in the bar data, the pixel value of images in MNIST ranges from 0 to 1 with 0 denoting blank space. Figure (4.2b) contains a representative sample of data from the MNIST data set.



(a) A random sampling of the generated bar data. This data contains 17% random uniformly distributed additive noise on top of the 'basis' images



(b) A random sampling of the MNIST data

Figure 4.2: Representative samples of both example data sets. The left panel contains simulated data while the right panel contains data collected from writing samples.

In both examples, we set up a network with only three total layers. The first layer is the input layer. Its size is governed by the size of the image being presented. The third layer is the output layer and its size is governed by the number of labels in the data. The second layer is the only hidden layer. In the following examples we explore the dependence of the Action and the results on the hidden layer size, $l_H(1)$. For simplicity, because the input and ouput layer sizes are

fixed by the structure of the data, and because we are only working with one hidden layer we will drop the layer number argument in the size of the hidden layer such that $l_H(1) \to l_H$.

Simulated Bar Images

The data plotted in Figure (4.2a) is very simple. A complete subset of data must contain at minimum 4 images, one of each type, to span the space determined by the 'basis' images. Additionally, because of the centered nature of the images, it is sufficient to know the value of only 3 pixels to be able to properly categorize an image. A noiseless image could be labeled appropriately if, for example, only the top left, top center, and top right pixel values were known. If the top right pixel had a value of 1 it would imply that the label should be $(0,1,0,0)$, if the top center pixel had a value of 1 it would imply that the label should be $(0,0,1,0)$, if the top left pixel had a value of 1 it would imply that the label should be $(0,0,0,1)$, and if all of these three pixels had a value of 0 it would imply that the label should be $(1,0,0,0)$. If we have some level of noise in the image generation, there is some threshold value that must be used instead of 1 to determine the label. For example, if none of the top pixels are above the noise level, then we know that the image is of a horizontal bar.

With such a simple set of images, feature space may be fully described in 3 dimensions though there may be several unique transformations from image to feature space that are adequate to properly characterize all images. If the minimum size of the network is correlated with the dimensionality of the feature space which characterizes the set of images being studied, we would assume that there would be a quantitative difference between the predictive capabilities of a network with $l_H < 3$ and $l_H > 3$.

At each hidden layer size we optimized a network on 10 of input-output pairs. The value of the Action at the largest R_f value is plotted in Figure (4.3). As expected there is a qualitative difference between the behavior of the Action at values smaller than $l_H = 3$. Following the training stage, we fix the network with the weights found through VA and present this new fixed

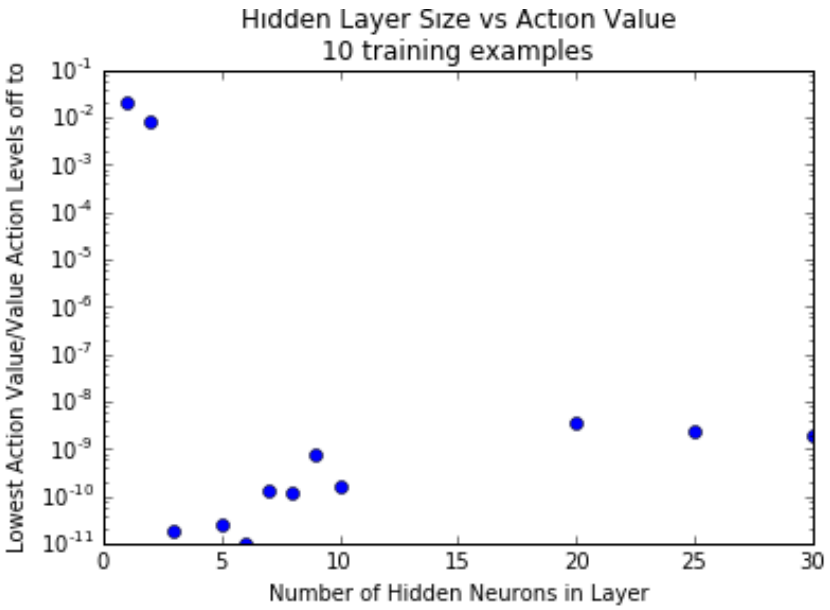


Figure 4.3: Analysis of Bar Image data with $M = 10$ and variable l_H. The Action drops significantly in magnitude when there are 3 neurons in the hidden layer. This implies that when $l_H = 3$ we are able to find a set of $\mathbf{x}$'s and $\overline{\mathbf{W}}$'s that generate an Action whose R_f term is zero and whose R_m term is dictated only by measurement noise. There is no significant difference in Action value for networks with hidden layer sizes between $l_H = 3$ and $l_H > 3$

network with a test data subset. This data subset is drawn from the unused images in the original set. The labels are hidden from the network. The output of the network is compared to the labels to determine the error with which the estimated network labels the presented images.

It should be noted that during the training of the hidden network, R_m was set to 1. This value of R_m implies that when the model is able to perfectly estimate the appropriate label, and thus the R_f term is zero, the value of the Action is expected to be equivalent to the average measurement noise level in each image. This value should be the variance of a uniform distribution of width 0.17, or on the order of 0.01. In Figure (4.3) you see that at $l_H = 3$ the value of the Action at the end of the training window drops from the expected value of 0.01 by several orders of magnitude. We can show that this drastic drop in the Action value corresponds to a drop in prediction error.

To compare the predicted label of new data with the true label, it is helpful to think of the

predicted label as a probability. As there is nothing in the structure of the network that constrains the sum of the node values in the output layer to be normalized, we apply a normalization function in the form of an adjusted softmax function:

$$x'_j = \frac{e^{x_j(l_F)} - 1}{\sum_j (e^{x_j(l_F)} - 1)} \tag{4.19}$$

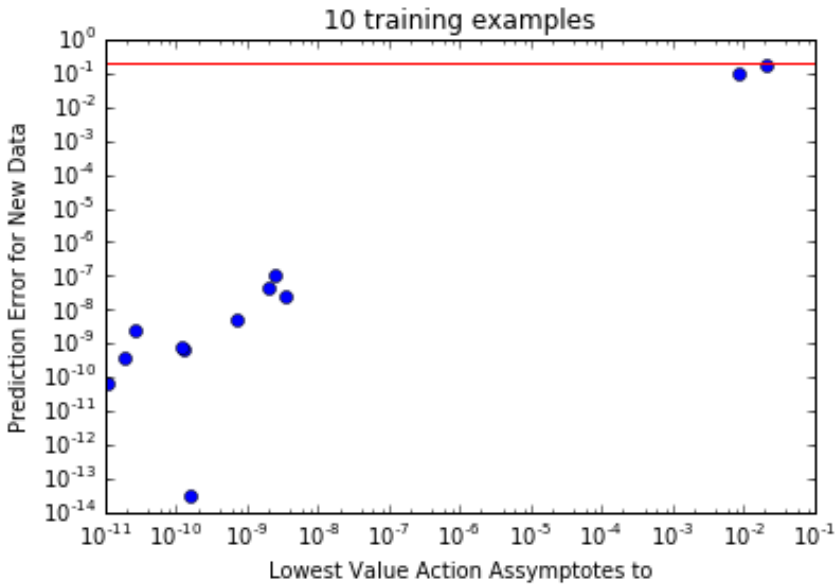
The error is then calculated by taking a least squares error between the normalized output of the network and the true label. This is averaged over a large number of test examples:

$$PE_{LS} = \frac{1}{M_{test} \cdot l_H(l_F)} \sum_{m}^{M_{test}} \left(\mathbf{x}'^{(m)} - \mathbf{y}^{(m)} \right)^2 \tag{4.20}$$

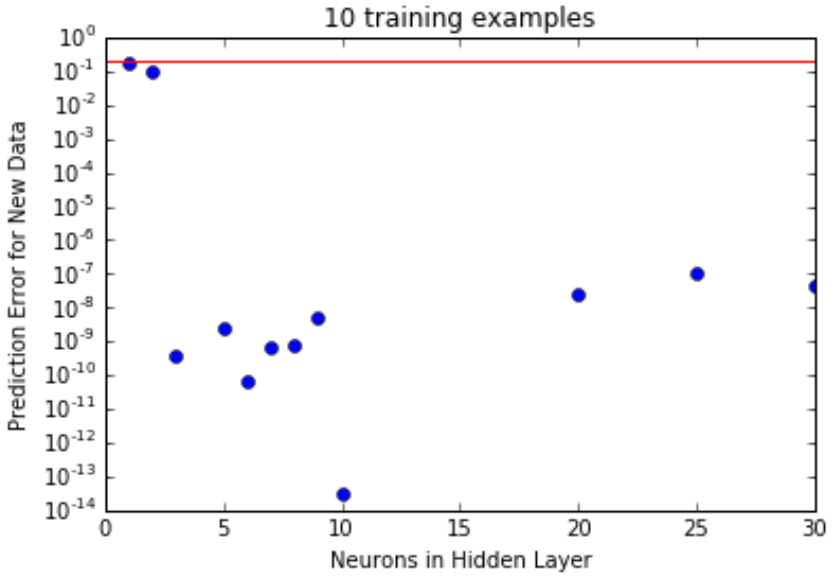
The $\frac{1}{l_H(l_F)}$ term is simply a scaling term over the size of the output layer. In all the plots demonstrating prediction error there is a red line which indicates the expected error in prediction if the weights in the network are randomly guessed.

Analysis of Figure (4.4) shows that the predictive capabilities of a network is highly correlated to the Action value that this network produced. The relationship between Action and prediction error appears approximately linear. Once the size of the network exceeds the threshold value at which the Action drops, there is very little change in the value of the Action with increasing size of hidden layer. Qualitatively the behavior of the prediction error as a function of hidden layer size is the same as the behavior of the Action. The prediction error does not significantly change as the size of the hidden layer increases past the threshold value at which the network is capable of characterizing the full system.

It is important to note that the calculations leading to Figure (4.3) and Figure (4.4) used $M = 10$ image-label pairs. Thus each image 'type' had at least two representatives within the training data. When $M = 2$ and not every image 'type' is represented in the training data, we get the plots shown in Figure (4.5). In this calculation, the Action appears qualititively to behave the same way as it does when $M = 10$. Naively, one might assume that the prediction plot for $M = 2$

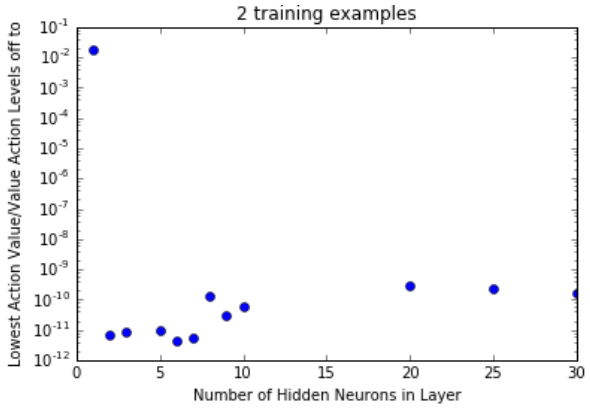


(a) Least Squares Prediction Error vs Action. High Action leads to high prediction error while low Action leads to low prediction error

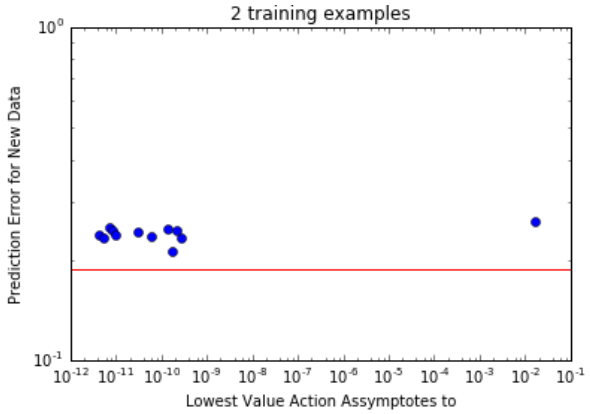


(b) Least Squares Prediction Error vs Layer size. There is a very severe and distinct drop in prediction error as the layer size exceeds the predicted necessary size of the hidden layer (3 neurons)

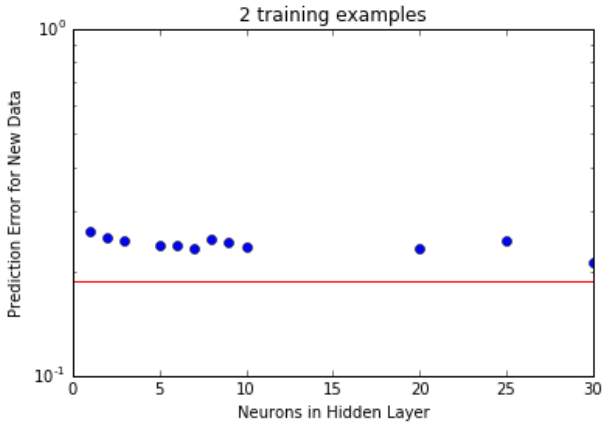
Figure 4.4: Analysis of Bar Image data with $M = 10$ and varying l_H. The relationship between Action and least squares prediction error suggests that the Action may be used as a metric for deciding the appropriate network size. Error is calculated using a least squares method after a modified softmax is applied to the output of the network. A horizontal line shows the expected error from a randomly guessed network.



(a) Qualitatively similar plot to Figure (4.3) of the Action versus hidden layer size where the machine was presented with 10 example images.



(b) Unlike the $M = 10$ case, the $M = 2$ vase shown here has no Variation in Error as Action changes



(c) In the $M = 2$ case here the network size has no effect on error, unlike in the $M = 10$ case.

Figure 4.5: Analysis of Bar Image data using 2 example images and varying the number of hidden nodes. These example images did not cover all of the 'types' of images that we wish to be able to characterize. Horizontal red line represents the expected prediction error for random guessing. The Action value plot does not look different from the $M = 10$ case shown in Figure (4.3), but the prediction never drops below random guessing.

would similarly mimic the prediction plot for $M = 10$. This assumption is incorrect. The $M = 2$ Action approaches the same assymptote as the $M = 10$ because the network is able to tune the weights to fit the two training examples perfectly. If the network is then tested through predictions on images that only fall into the 'types' on which this network was trained, the prediction error would follow the same pattern as in the $M = 10$ example. The large prediction errors in the $M = 2$ example appear when trying to characterize an image type that the network has not yet seen. Thus it is vital when training a network on data that the full space of the data is presented to the network while estimating the weight values. Choosing an appropriate amount of data is a responsibility left to the researcher as the network itself can not be used to inform whether feature space is sufficiently represented. A simple test in the supervised learning regime is to make sure that the training data consists of at least one or more of each type of image and that the labels in the training data consist of all possible labels.

MNIST Data Set

The second set of data that VA inspired machine learning methods were applied to is the Modified National Institute of Standards and Technologies (MNIST) database. This is a set of handwritten digits collected from American Census Bureau employees and American high school students. The original data set used the Census employees' samples as a training set and the high school students' samples as a test or validation set. This was unsuccessful as it appears that there is more variation in high school students' handwriting than in the handwriting of government employees. The creators of the MNIST data set took the unmodified set and reshuffled the samples [25].

There are many different neural network structures and methods for minimizing the cost function in the field of machine learning. A common thread among neural network training done using MNIST data is that the full 60,000 image training set is used to tune the parameters of the network. The tuned network is then tested on 10,000 test images. Such analysis typically

acheives an error on the order of 1%. The error here is calculated by counting the number of times the tuned neural network assigns an incorrect label to an image. Many methods of training neural networks on MNIST data additionally take advantage of preprocessing techniques which improve the quality of the characterization.

The analysis presented in this thesis will not train on the full training set of 60,000 images. Additionally, the analysis presented here includes no preprocessing or filtering of the images. Due to this, the results are not expected to perform at the same level as the more successful current methods. Instead the following results demonstrate a new metric, the Action, which may be used to tailor the structure of the neural network and determine the optimal network for prediction prior to running any predictions. The ability to adapt the structure of the network through the use of the Action is desireable in future work where the labels for new data for which predictions are generated are unknown to the researcher.

As mentioned previously, when the network model constraints are enforced through the use of VA and the weight estimates approach the correct values, the Action approaches a value given by the expectation value of the R_m term of Eq. (4.18). For computational ease, we set $R_m = 1$ and thus expect the Action to approach a value equal to $\frac{1}{2}\langle(\mathbf{x}(0) - \mathbf{y}_{input})^2 + (\mathbf{x}(l_F) - \mathbf{y}_{output})^2\rangle = \frac{\sigma^2}{2}$ where σ is the noise in the image and label data. When choosing the weight matrix, $\overline{\mathbf{W}}$, to generate predictions, we utilize Laplace's method by choosing the $\overline{\mathbf{W}}$ that produces the lowest Action and thus the highest probability. This method requires that the lowest Action level be significantly smaller than the next lowest to be valid. For this reason it is very important to know the expected noise value, σ, so that we may estimate the expected value of the Action and determine whether there is sufficient separation between Action levels to use Laplace's method to estimate $\overline{\mathbf{W}}$.

Because the MNIST images are real images of handwritten digits it is difficult to determine the value of σ. The measurement noise that arises from error in the scan is minimal. This can be confirmed by determining the variance of the pixel value in the 'blank' regions of the image. In addition to the noise incured from copying the images, there is some noise due to the variation in

a specific digit drawn by different people. Not only is there variation in the shape and orientation, but the width of the strokes may vary based on the pens used and the pressure applied. This type of noise is difficult to characterize quantitatively. At first approximation we characterize the noise by taking a histogram of pixel values present in the images and approximating the width of the peak at zero. In a sense, this is the approximation of the additive noise on the blank regions of the images.

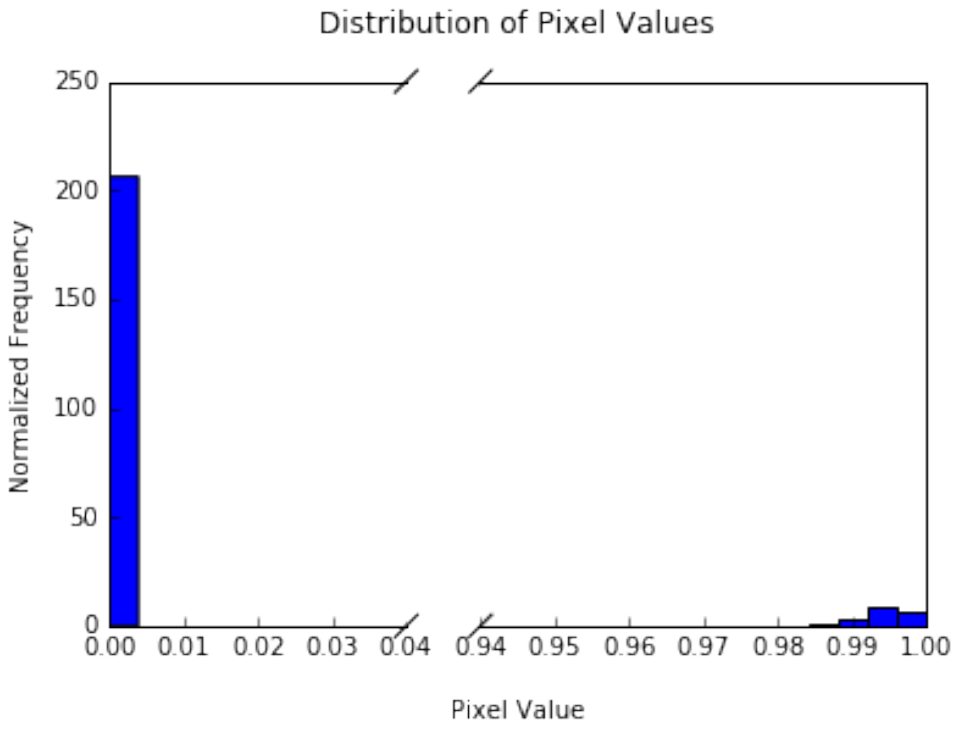


Figure 4.6: Histogram of all pixel values in MNIST data set. The maximum near pixel value of zero represents the blank regions of the images while the maximum near pixel value one represents the regions in which the digit is drawn.

There are two maxima in the histogram shown in Figure (4.6). The first occurs near pixel value zero while the second occurs near pixel value 1. The width of the first maximum represents the variation in the pixel value on blank regions of an image. this can be used to estimate the additive noise incured during the collection of these images. As demonstrated in Figure (4.6) this additive noise is particularly small. Additionally, the output data or labels have exactly zero noise

as they are perfectly assigned to the images. This combination of factors leads to a very small, approaching zero, expectation value for the Action. This is demonstrated in the results presented in the following seection.

MNIST - 20 Examples - all 10 digits

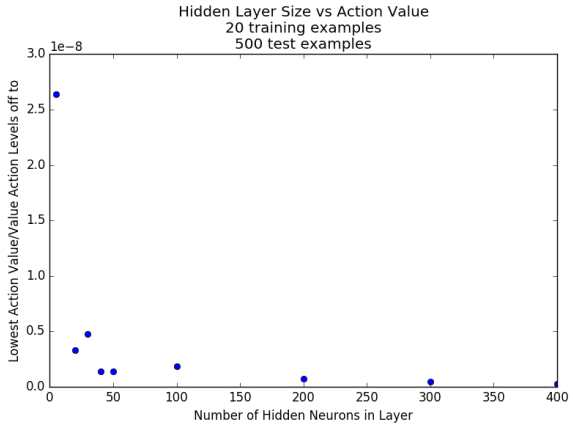
We first test the VA based machine learning procedure using $M = 20$ image-label pairs. This is an insufficient number of examples to characterize the full set of digits as there are on average only two versions of each digit. Prediction error was calculated using the expression shown in Eq. (4.20) with $M_{test} = 500$. The test data used for predictions was approximately evenly distributed around all 10 possible digits. The resultant plots are shown in Figure (4.7).

Though there is some decrease in the prediction error as a function of layer size, there does not exist the same drastic change in prediction error with Action level decrease as is seen in Figure (4.4). The weak response in the prediction plot is most likely due to a insufficient quantitiy of information presented to the network, analogous to the $M = 2$ bar data example in which not all types of bar images in the previous section were presented.

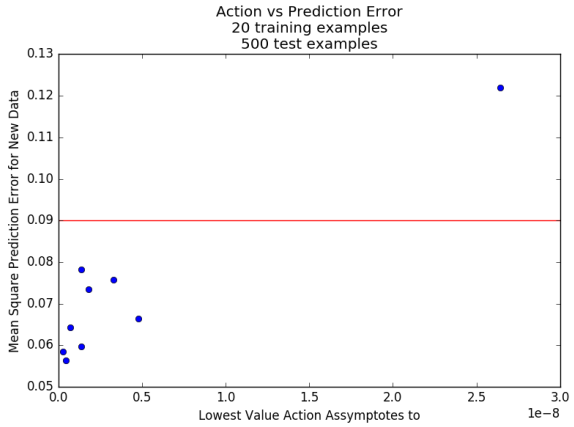
MNIST - 100 examples - 2 digits

We now focus on a subset of the MNIST data that we call MNIST-lite. This data consists of images of only 1's and 7's. These particular digits were chosen because of their visual similarity. The decrease in dimensionality of the output data allows the network used for the training to contain fewer weights and decreases the computational difficulty of the problem. In addition to looking at a subset of the data, we have adjusted the data in such a way that the mean pixel value of each image is zero. This was done primarily as a method to stabalize the numerical results of the VA method due to the chosen form of $\mathbf{f}_{activation}$, a sigmoid function centered at zero. As a secondary result, this shift in pixel value causes the distribution over all images around the minimum pixel value to widen without changing the distribution over each individual image.

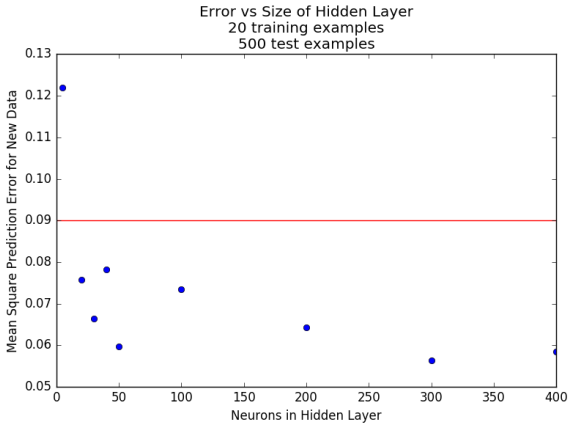
We present a varying M number of examples of 1's and 7's to the neural network with



(a) Action as a function of l_H for MNIST data with $M = 20$. There is a drop in Action as the hidden layer increases



(b) Prediction error as a function of Action for MNIST data with $M = 20$. Prediction error and Action level appear to be highly correlated.



(c) Prediction error as a function of l_H with $M = 20$. Error drops slightly below random guessing with layer size increase. High error is probably due to low M value

Figure 4.7: 20 examples for MNIST training, varying hidden layer size, all 10 digits presented to network for analysis. Prediction error averaged over 500 test examples. Horizontal red line represents error expected if predictions are made through random guessing.

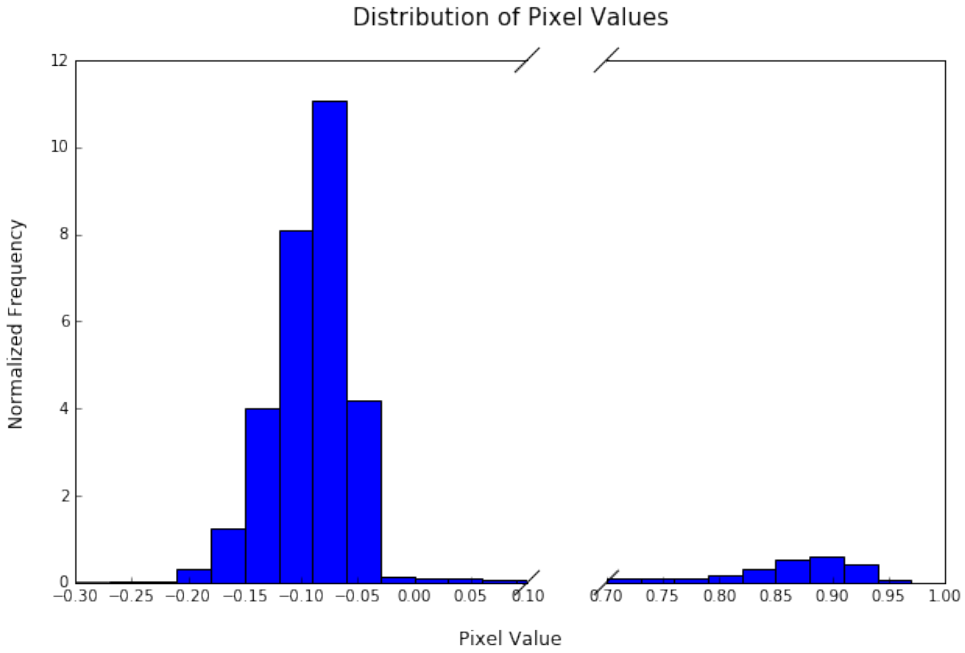


Figure 4.8: Distribution of pixel values for MNIST-lite. The mean pixel value of each image was subtracted from the image data of the corresponding image to generate a set of image pixel matrices with zero mean. This shift widens the distribution of pixel values in the blank regions of the images without changing the information contained in the data set.

one hidden layer and once again vary the size of the hidden layer. Both the Action and error are plotted as a function of the hidden layer size. In this experiment, the prediction error was calculated as a binary classification error:

$$PE = \frac{1}{M_{test}} \sum_{m}^{M_{test}} PE_m$$
$$PE_m = \begin{cases} 0 & \text{if } \operatorname{argmax}(\mathbf{y}_m) = \operatorname{argmax}(\mathbf{x}_m) \\ 1 & \text{if } \operatorname{argmax}(\mathbf{y}_m) \neq \operatorname{argmax}(\mathbf{x}_m) \end{cases} \tag{4.21}$$

With a single hidden layer, networks of all sizes produce very little variation in Action level. The left hand plot in Figure (4.9) shows the mean Action level as a function of layer size and M. Though the trend in Action as a function of layer size is as expected, there is no clear separation between the Action levels to indicate which must be used in future predictions. As

such, predictions were done on all the estimated networks and the mean prediction error was plotted. The right hand plot in Figure (4.9) shows the results of this prediction analysis.

As expected the prediction error drops as the Action level drops with increasing hidden layer size and amount of training data. Surprisingly, the lowest prediction error path does not always correspond to the lowest Action level path. This may be explained by the fact that all of the Action levels are extremely close in magnitude and Laplace's approximation is no longer accurate.

Additionally, the lowest Action levels are smaller in magnitude when compared to the expected value of the Action for a good estimate. One estimate of the expected Action value can be made by estimating the variance of the lower peak in Figure (4.8) which is approximately 10^{-2}. Examination of Figure (4.9) shows that networks producing Action values on the order of the expected value predict with high error, but paths in which the Action is significantly lower than the expected predict with high accuracy. This is consistent with the results from the Bar Image data shown in Figures (4.3) and (4.4).

Regardless of the issues mentioned above, the prediction accuracy for all estimates are comprable to those found in previous analysis using $M = 60,000$ and significantly more complex network structures. Vareational Annealing ML, at least in this simplified example of 2 digit classification, performs on par with previously designed single hidden layer neural networks which produce errors between 0.7% and 4.7% depending on the size, connectivity, and specific method of analysis [26]. These previous results were achieved using 60,000 training examples for all ten digits. This is 120 times the number of examples per image type as compared to the analysis done in this thesis. Additionally, all previous results were accomplished with the number of hidden neurons given by $l_H \geq 300$. In the results presented above, the mean prediction error reaches a minimum values at around $L_H \approx 10$ for nearly all M values used. Though this may in part be due to the lower number of digit types presented to the network, both the Action and prediction error plots demonstrate that there is no value in arbitrarily increasing the size of

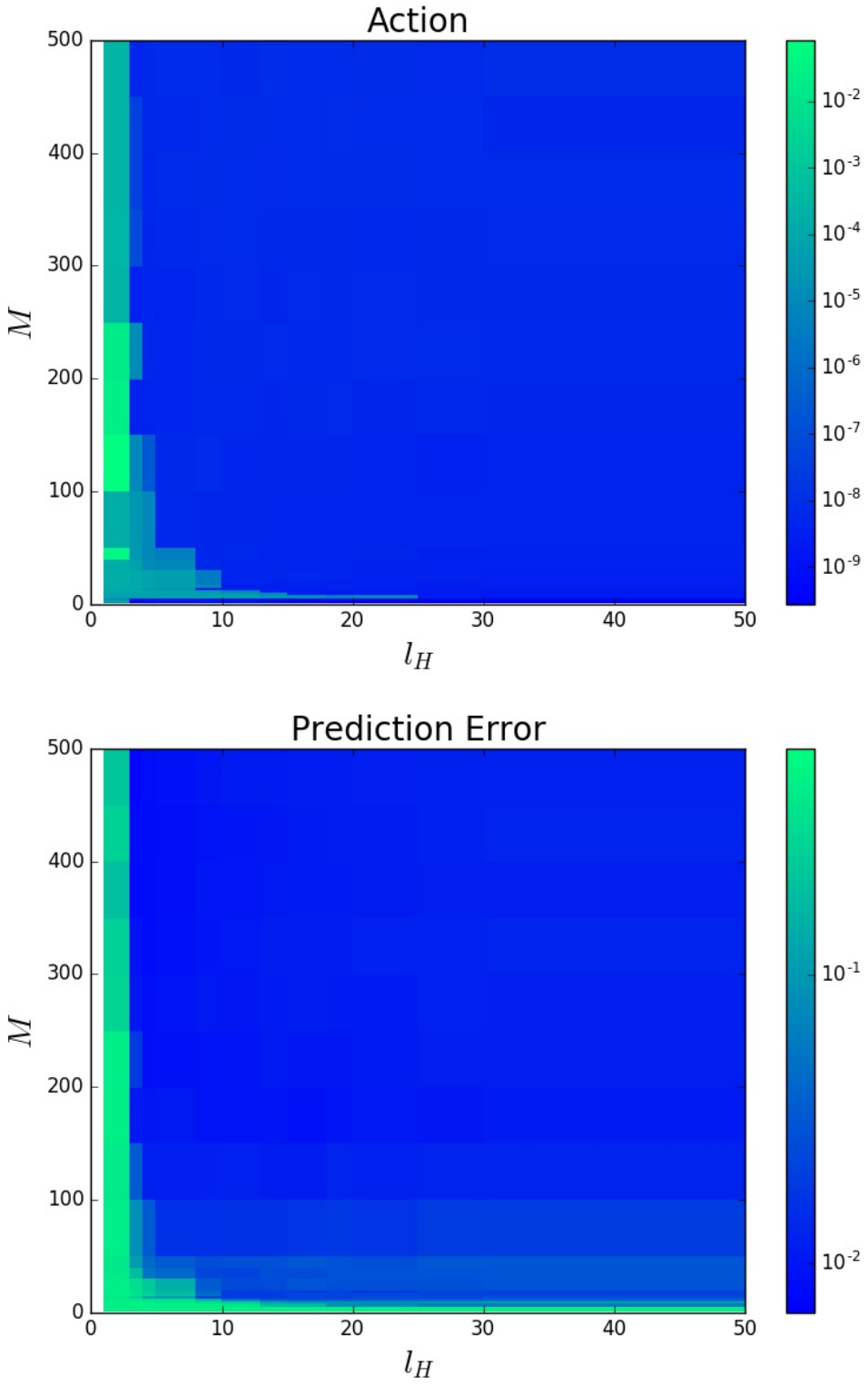


Figure 4.9: Action and Prediction Error plotted as a function of hidden layer size. 100 training examples of 1's and 7's. M_{test} =10,000 prediction test examples. Single hidden layer.

the network and the Action may be used while learning the full MNIST data set as a metric to determine the optimal network size.

4.4 Discussion

The field of machine learning aims to solve a problem analogous to one which has been well studied in dynamical systems. The arbitrary nature of the chosen networks in machine learning problems necessitates the model error approximation that has been developed and utilized in the estimation of states and parameters in dynamical system problems. The use of a model error term in the estimation of the probability density function when searching for parameter and state values in dynamical systems has been shown to improve estimates. The gradual enforcement of model constraints on the neural network training set similarly allows for success in the discrete machine learning problem.

In addition to success in training a substantially smaller set of data with comparable predictability to previous results, the Action can be used as a metric to determine the quality of prediction and the minimum necessary size of the network. Due to the mathematical definition of the Action and from insight gained through dynamical systems research, we had expected the Action in a ML problem to approach the measurement error for networks which predicted future image labels accurately. Instead, Figure (4.9) demonstrates that a significant drop from the $A \approx \sigma^2$ corresponds to low prediction error. The difference in the relationship between Action value and prediction error between dynamical systems and ML problems is unclear and may be a direction for future research.

In the case of the simple bar graphs in which the necessary size of the hidden layer can easily be approximated by hand as the number of pixels whose value must be known to properly categorize the image, the Action and predictive capability of the network behave as expected: both decreasing dramatically as the size of the network exceeds the necessary size. For the more

complicated data set of MNIST images, the correlation between prediction quality and Action value persists and the relationship between Action value and the dimension of the hidden layer may be used to gain insight into the dimension of feature space of images in the training set.

Due to the correlation between the prediction quality and the Action value, further research may explore additional methods through which the Action may be used as a metric to construct and design the appropriate neural network for a given set of data.

This chapter has been adapted from the material submitted for publication in H. D. I. Abarbanel, P. J. Rozdeba, and S. Shirman, Machine Learning; Deepest Learning as Statistical Data Assimilation Problems, Neural Computation and contains work that will be prepared for publication in the future in H.D.I. Abarbanel, G. Silva,S. Shirman, V. George, A. Gupta, Z. Fang, P. Rozdeba, A. Ty and R. Gonzalez with permission from all authors and collaborators. The dissertation author was a coauthor and collaborator in both works.

Appendix A

Final Notes

This work approached the question of transferring information from collected measurements through a mathematical model of the system to unknown state trajectories and parameter values with the ultimate goal to characterize the model and successfully predict the future trajectory of all states in the system of study. Methods of information transfer can be applied across many fields to problems in which the goal is to extract information from available data to unmeasured quantities that are coupled to the recorded quantities through some mathematical expression. Two such problems, the estimation of parameters and states in a dynamical system and the estimation of machine weights necessary to accurately label categories of images, were presented in this thesis demonstrating the broad application of these methods.

On the topic of dynamical systems parameter and state estimation, a computational method called SMC was presented to address the shortcomings of prior DA methods including strong constraint methods such as MLEF and Kalman filtering and optimization methods such as VA [39, 16, 37]. The model error term in the Action, adapted from VA methods, accounts for stochastic variation in the time evolution of model states as well as errors in the mathematical formulation of the model while the use of a random walk around regions of high likelihood extract additional information not gained through optimization methods about the structure of the

probability distribution and the error in estimates made through the optimization DA methods. Though previous work applying Monte Carlo methods to sampling dynamical systems while allowing for model error generates acceptable estimates of parameter and state trajectories and their higher moments [21], attempting a random search over all state and parameter space becomes computationally intractable as the dimensionality of the problem increases. Informing and constraining the search to regions suggested through optimization methods while simultaneously using VA optimization methods to determine the correct level of model constraint addresses the shortcomings of DA methods with strong model constraints, optimization methods, and random walks which search over the full state and parameter space.

A random search over a high dimensional space, particularly when the probability density function is correlated in a nonlinear manner will be less stable over short searches than a lower dimension and simpler system. Due to this, and the effect of importance sampling on decreasing the effective number of samples in a search [29], we must either ensure that a random walk over a probability density function of interest contains a sufficient number of samples or that the method through which the region is sampled is optimized. More efficient sampling methods for high dimensional spaces may be employed to address the issue of instability in the random search when using SMC in future work in which the dimensionality of the problem has increased. This will be particularly relevant when studying a more biologically realistic neural model with a larger number of ion channels and a longer time series of data. Such improved sampling methods may include the Metropolis-adjusted Langevin algorithm [34], Hamiltonian Monte Carlo methods [9], or other adaptive random walk methods which drastically increase the rate at which the approximated distribution converges to the sampled one [32].

The application of SMC to a neuron toy model produces interesting insight into the system. In addition to discovering a new local maximum for the probability distribution which predicts the future membrane voltage time series with a high level of accuracy, SMC analysis of the data confirms assumptions made in previous work [35, 5, 18] about the structure of the

probability distribution function. The distribution estimated by SMC analysis of NaKL simulated data contains very rapidly decaying probability maxima in keeping with the structure of the probability density function necessary for an accurate use of Laplace's method. This search confirms that optimization of the probability is an appropriate method for estimating parameters in an NaKL neuron.

Further work on more complicated neurons can confirm this conclusion for other HH neuron types. Besides confirming previous assumptions about the structure of the distribution, SMC may be used in future work to study the effect of data quantity and quality on the probability distribution. By applying SMC analysis to twin experiments with varying driving currents, quantities of data, and noise levels, the necessary amount of data to best estimate the parameters and states of the system can be determined. Adjusting the data and experimental design in twin experiments until the estimated probability distribution contains the fewest number of local maxima which decay rapidly and overlap with the parameters and states used to generate the data may be used to inform the design of biological experiments to best utilize data collected from neurons.

The second problem addressed in this thesis resulted in the application of DA methods to ML problems. This work shows promise in streamlining the process of learning in a machine. Determining the cost function through a generalization of the dynamical systems Action accounts for both the quality of the data presented as well as the likelihood of the chosen transformation function being an accurate one allows for the gradual enforcement of the mathematical model determined by the chosen neural network structure. This gradual enforcement of the model constraints allows for acceptable weight estimates that produce accurate predictions on new data with less training and smaller neural networks than in previous ML work [26].

Given the correlation between prediction error and Action value demonstrated in Chapter 4, future work will focus on applying information gained from the Action due to VA to optimize the structure of the neural network conditioned on the data provided. In other words, DA

methods may be used to determine the most computationally efficient artificial neural network needed to extract the maximum amount of information from available data. This will reduce the computational strain required to analyze and train the machine while also providing insight into the minimal mathematical transformation from input data to label which may in turn provide information on the basis features of the data.

References

[1] Henry D. I. Abarbanel. *Predicting the future: completing models of observed complex systems*. Springer, 2013.

[2] Henry D. I. Abarbanel, Paul Rozdeba, and Sasha Shirman. Machine learning, deepest learning: Statistical data assimilation problems. *CoRR*, abs/1707.01415, 2017.

[3] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. Ion channels and the electrical properties of membranes. 2002.

[4] Eve Armstrong, Amol V. Patwardhan, Lucas Johns, Chad T. Kishimoto, Henry D. I. Abarbanel, and George M. Fuller. An optimization-based approach to calculating neutrino flavor evolution. *Phys. Rev. D*, 96:083008, Oct 2017.

[5] Daniel Breen, Sasha Shirman, Eve Armstrong, Nirag Kadakia, and Henry Abarbanel. Hvc interneuron properties from statistical data assimilation. *arXiv preprint arXiv:1608.04433*, 2016.

[6] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[7] Li Deng and Dong Yu. Deep learning: Methods and applications. Technical report, May 2014.

[8] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

[9] Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216 – 222, 1987.

[10] J-P Eckmann and David Ruelle. Ergodic theory of chaos and strange attractors. In *The Theory of Chaotic Attractors*, pages 273–312. Springer, 1985.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA; London, UK, 2016.

[12] Celso Grebogi, Edward Ott, and James A. Yorke. Chaos, strange attractors, and fractal basin boundaries in nonlinear dynamics. *Science*, 238(4827):632–638, 1987.

[13] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[14] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

[15] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[16] Brian R. Hunt, Eric J. Kostelich, and Istvan Szunyogh. Efficient data assimilation for spatiotemporal chaos: A local ensemble transform kalman filter. *Physica D: Nonlinear Phenomena*, 230(1):112 – 126, 2007. Data Assimilation.

[17] Daniel Johnston and Samuel Miao-Sin Wu. *Foundations of Cellular Neurophysiology*. Bradford Books, MIT Press, 1995.

[18] Nirag Kadakia, Eve Armstrong, Daniel Breen, Uriel Morone, Arij Daou, Daniel Margoliash, and Henry DI Abarbanel. Nonlinear statistical data assimilation for HVC_{RA} neurons in the avian song system. *Biological cybernetics*, 110(6):417–434, 2016.

[19] A Karimi and Mark R Paul. Extensive chaos in the lorenz-96 model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(4):043105, 2010.

[20] Mark Kostuk. Synchronization and statistical methods for the data assimilation of hvc neuron models. *PhD Dissertation in Physics, University of California, San Diego*, 2012.

[21] Mark Kostuk, BryanA. Toth, C.Daniel Meliza, Daniel Margoliash, and HenryD.I. Abarbanel. Dynamical estimation of neuron and network properties ii: path integral monte carlo methods. *Biological Cybernetics*, 106(3):155–167, 2012.

[22] William A Lahoz and Philipp Schneider. Data assimilation: making sense of earth observation. *Frontiers in Environmental Science*, 2:16, 2014.

[23] Pierre Simon Laplace. Memoir on the probability of causes of events. *Mémoires de Mathématique et de Physique, Tome Sixième*, pages 621–656, 1774.

[24] Pierre Simon Laplace. Memoir on the probability of the causes of events. *Statistical Science*, 1(3):364–378, 1986. Translation to English by S. M. Stigler.

[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[26] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database. `http://yann.lecun.com/exdb/mnist/`.

[27] Andrew C Lorenc and Tim Payne. 4d-var and the butterfly effect: Statistical four-dimensional data assimilation for a wide range of scales. *Quarterly Journal of the Royal Meteorological Society*, 133(624):607–614, 2007.

[28] Edward N. Lorenz and Kerry A. Emanuel. Optimal sites for supplementary weather observations: Simulation with a small model. *Journal of the Atmospheric Sciences*, 55(3):399–414, 1998.

[29] Luca Martino, Víctor Elvira, and Francisco Louzada. Effective sample size for importance sampling based on discrepancy measures. *Signal Processing*, 131:386–401, 2017.

[30] Jonathan C. Mattingly, Natesh S. Pillai, and Andrew M. Stuart. Diffusion limits of the random walk metropolis algorithm in high dimensions. *Ann. Appl. Probab.*, 22(3):881–930, 06 2012.

[31] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, 21:1087–1092, June 1953.

[32] Matthias Morzfeld, Xin T Tong, and Youssef M Marzouk. Localization for mcmc: sampling high-dimensional posterior distributions with banded structure. *arXiv preprint arXiv:1710.07747*, 2017.

[33] K. G. Murty and S. N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987.

[34] Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

[35] Bryan A Toth, Mark Kostuk, C Daniel Meliza, Daniel Margoliash, and Henry D I Abarbanel. Dynamical estimation of neuron and network properties i: variational methods. *Biological Cybernetics*, 105(3-4):217–237, 2011.

[36] A. Wachter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[37] J. Ye, N. Kadakia, P. J. Rozdeba, H. D. I. Abarbanel, and J. C. Quinn. Improved variational methods in statistical data assimilation. *Nonlinear Processes in Geophysics*, 22(2):205–213, 2014.

[38] J. Ye, Daniel Rey, Nirag Kadakia, Michael Eldridge, Uri Morone, Paul Rozdeba, Henry D. I. Abarbanel, and John C. Quinn. A systematic variational method for statistical nonlinear state and parameter estimation. *Physical Review E*, 2015.

[39] Milija Zupanski. Maximum likelihood ensemble filter: Theoretical aspects. *Monthly Weather Review*, 133(6):1710–1726, 2005.