

AFIT/GE/ENG/16-..

ANISOTROPIC SIMULATED ANNEALING AND ITS APPLICATION TO FEED
FORWARD NEURAL NETWORK WEIGHT SELECTION

THESIS

Justin Fletcher
First Lieutenant, USAF

AFIT/GE/ENG/16-..

Approved for public release; distribution unlimited

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

AFIT/GE/ENG/16-..

ANISOTROPIC SIMULATED ANNEALING AND ITS APPLICATION TO
FEED FORWARD NEURAL NETWORK WEIGHT SELECTION

THESIS

Presented to the Faculty of the Electrical and Computer Engineering
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computer Science

Justin Fletcher, B.S. CEC

First Lieutenant, USAF

June, 2016

Approved for public release; distribution unlimited

AFIT/GE/ENG/16-..

ANISOTROPIC SIMULATED ANNEALING AND ITS APPLICATION TO
FEED FORWARD NEURAL NETWORK WEIGHT SELECTION

Justin Fletcher, B.S. CEC

First Lieutenant, USAF

Approved:

Dr. Michael J. Mendenhall
Thesis Advisor

Date

Dr. Gilbert L. Peterson
Committee Member

Date

Capt. Charlton D. Lewis

Date

Ph.D.
Committee Member

Date

Preface

Justin Fletcher

Table of Contents

	Page
Preface	iii
List of Figures	vii
List of Tables	xi
List of Symbols	xii
List of Abbreviations	xiii
Abstract	xiv
I. Introduction	1-1
II. Background	2-1
2.1 Artificial Neural Networks	2-1
2.1.1 Biological Inspiration	2-1
2.1.2 Historical Overview	2-2
2.1.3 Network Topology	2-11
2.1.4 Activation Functions	2-11
2.1.5 Learning Strategies	2-11
2.2 Related Works in Simulated Annealing	2-11
2.2.1 Reannealing	2-14
2.3 Related Works in Quantum Mechanics	2-14
2.3.1 Quantum Tunneling	2-14
2.4 Notation and Terminology Conventions	2-15

	Page
III. Methodology	3-1
3.1 Simulated Annealing	3-1
3.1.1 Neighborhood Functions: Traversing the Cost Surface	3-1
3.2 Feed-Forward Neural Network Representation	3-10
3.3 Applying Simulated Annealing to Feed-Forward Neural Net- work Weight Modification	3-13
3.3.1 Synaptic Annealing Neighborhood Functions	3-13
3.3.2 Feed-Forward Neural Network Anisotropy Policies	3-15
3.4 Weight Space Traversal	3-15
3.4.1 Gaussian Neighborhood Function Weight Space Traver- sal	3-16
3.4.2 Cauchy Neighborhood Function Weight Space Traversal	3-17
3.4.3 Isotropic GSA Neighborhood Function Weight Space Traversal	3-19
3.4.4 Weight-Anisotropic GSA Neighborhood Function Weight Space Traversal	3-20
3.4.5 Uniform Neighborhood Function Weight Space Traver- sal	3-22
3.4.6 Comparative Weight Space Traversal	3-22
3.5 Cost Functions	3-25
3.5.1 Regression Error Cost Function	3-25
3.5.2 Classification Error Cost Function	3-26
3.6 Data	3-26
3.6.1 Classification Data Sets	3-26
3.6.2 Function Approximation Data Sets	3-26
3.7 Design of Experiments	3-27
IV. Results	4-1
4.0.1 Classification Performance	4-1
4.0.2 Function Approximation Performance	4-1

	Page
V. Conclusion	5-1
Appendix A. First appendix title	A-1
A.1 In an appendix	A-1
Bibliography	BIB-1
Vita	VITA-1

List of Figures

Figure		Page
2.1.	A simple perceptron.	2-4
2.2.	(Bottom) A simple step potential in one dimension. (Top) The probability density function of a generic quantum system in the presence of a potential energy step barrier. There is an exponential decrease in probability through the barrier, and a uniform probability beyond the barrier.	2-15
3.1.	(a) A potential energy barrier on a one-dimensional potential energy surface. (b) The tunneling probability relative to the probability of traversal due to thermal fluctuation for a step barrier plotted as a function of the height and width of the barrier.	3-4
3.2.	This figure comprises four plots, each of which displays a GSA distribution at various values of T_{q_V} , in order to illustrate the behavior of the GSA distribution for several values of q_V . This figure shows the GSA near the mean, which illustrates the effect of q_v and T_{q_V} on the near-mean domain values, while neglecting the effects on the tails of the distribution.	3-6
3.3.	This figure comprises four plots, each of which displays a GSA distribution at various values of T_{q_V} , in order to illustrate the behavior of the GSA distribution for several values of q_V . This figure shows the GSA distribution for domain values far from the mean, which illustrates the effect of q_v and T_{q_V} on the tails of the distribution. .	3-8
3.4.	(a) An arbitrary feed-forward ANN. (b) The weight matrix representation of the feed-forward ANN in (a).	3-11

Figure		Page
3.5.	(a) A plot showing the traversal of the w_1, w_2 subspace of the weight space over the course of 5,000 epochs produced by a synaptic annealing algorithm employing a Gaussian visiting distribution. A solid line indicates a move which was accepted by the simulate annealing algorithm, while a dashed line indicates a move which was rejected. In this figure, only a few rejected moves are visible. The word <i>start</i> indicates the initial value of (w_1, w_2) , while the word <i>end</i> denotes the final value. (b) (upper) A plot showing the both the training and validation MSE of the results produced by the neural network in each epoch, smoothed using a central moving window average with a width of 21. This is the post-perturbation error, meaning that the error associated with moves that were rejected is shown. (lower) A plot showing the sum of squared weights of the neural network during each training epoch.	3-17
3.6.	(a) A plot showing the traversal of the w_1, w_2 subspace of the weight space over the course of 5,000 epochs produced by a synaptic annealing algorithm employing a Cauchy visiting distribution. A solid line indicates a move which was accepted by the simulate annealing algorithm, while a dashed line indicates a move which was rejected. In this figure, only a few rejected moves are visible. The word <i>start</i> indicates the initial value of (w_1, w_2) , while the word <i>end</i> denotes the final value. (b) (upper) A plot showing the both the training and validation MSE of the results produced by the neural network in each epoch, smoothed using a central moving window average with a width of 21. This is the post-perturbation error, meaning that the error associated with moves that were rejected is shown. (lower) A plot showing the sum of squared weights of the neural network during each training epoch.	3-18

- 3.7. (a) A plot showing the traversal of the w_1, w_2 subspace of the weight space over the course of 5,000 epochs produced by a synaptic annealing algorithm employing an isotropic GSA visiting distribution. A solid line indicates a move which was accepted by the simulate annealing algorithm, while a dashed line indicates a move which was rejected. In this figure, only a few rejected moves are visible. The word *start* indicates the initial value of (w_1, w_2) , while the word *end* denotes the final value. (b) (upper) A plot showing the both the training and validation MSE of the results produced by the neural network in each epoch, smoothed using a central moving window average with a width of 21. This is the post-perturbation error, meaning that the error associated with moves that were rejected is shown. (lower) A plot showing the sum of squared weights of the neural network during each training epoch. 3-20
- 3.8. (a) A plot showing the traversal of the w_1, w_2 subspace of the weight space over the course of 5,000 epochs produced by a synaptic annealing algorithm employing a GSA visiting distribution with synaptic weight-based anisotropy. A solid line indicates a move which was accepted by the simulate annealing algorithm, while a dashed line indicates a move which was rejected. In this figure, only a few rejected moves are visible. The word *start* indicates the initial value of (w_1, w_2) , while the word *end* denotes the final value. (b) (upper) A plot showing the both the training and validation MSE of the results produced by the neural network in each epoch, smoothed using a central moving window average with a width of 21. This is the post-perturbation error, meaning that the error associated with moves that were rejected is shown. (lower) A plot showing the sum of squared weights of the neural network during each training epoch. 3-21

Figure		Page
3.9.	(a) A plot showing the traversal of the (w_1, w_2) subspace of the weight space over the course of 5,000 epochs produced by a synaptic annealing algorithm employing a visiting distribution which is uniform over the range $[-\frac{1}{2}, \frac{1}{2}]$. A solid line indicates a move which was accepted by the simulate annealing algorithm, while a dashed line indicates a move which was rejected. In this figure, only a few rejected moves are visible. The word <i>start</i> indicates the initial value of (w_1, w_2) , while the word <i>end</i> denotes the final value. (b) (upper) A plot showing the both the training and validation MSE of the results produced by the neural network in each epoch, smoothed using a central moving window average with a width of 21. This is the post-perturbation error, meaning that the error associated with moves that were rejected is shown. (lower) A plot showing the sum of squared weights of the neural network during each training epoch.	3-23
3.10.	(a) A plot juxtaposing the weight space traversals of produced by a synaptic annealing algorithm employing several different visiting distributions.	3-24
3.11.	(a) A plot displaying the surface produced by the complicated interaction function, f_{CI} , in two dimensions, u_1 and u_2 (b) A color contour plot of the complicated interaction function.	3-27
4.1.	The simulation-time evolution of the 25-fold cross-validated classification error of a feed-forward neural network with 50 hidden units on Fishers iris data. Each column in this table depicts the results for an SA neighborhood function employing a single type of visiting distribution, which is indicated in the column header. Each row depicts a single type of anisotropy strategy, which is indicated in the row header. The classification error is defined as the fraction of samples incorrectly classified.	4-1

List of Tables

Table		Page
3.1.	Training and Validation Set Mean Squared Error at Epoch 5,000 .	3-25

List of Symbols

Symbol

Page

List of Abbreviations

Abbreviation	Page
ANN Artificial Neural Network	xiii
1	
Simulated Annealing	2-13
ANN	

AFIT/GE/ENG/16-..

Abstract

Stub.

ANISOTROPIC SIMULATED ANNEALING AND ITS APPLICATION TO FEED FORWARD NEURAL NETWORK WEIGHT SELECTION

I. Introduction

Stub.

II. Background

This chapter serves as a comprehensive review of the physical and computational concepts material to the topic of this thesis. A broad overview of artificial neural networks and the application and history thereof is presented. Next, various formulations of simulated annealing are described, along with a summary of some related works and a description of the physical inspiration for the algorithm. The chapter concludes with a very brief overview of the quantum mechanics, with emphasis placed on those concepts which will be employed throughout the document. Finally, the notation and terminology conventions adopted in this thesis are established.

2.1 Artificial Neural Networks

It has long been recognized that the capacity of biological information processing systems to flexibly and quickly process large quantities of data greatly exceeds that of sequential computing machinery. This information processing capability arises from the complex, nonlinear, parallel nature of biological information processors. The family of models designed to replicate this powerful information processing architecture are collectively called artificial neural networks (ANNs). In the most general sense, ANNs are parallel distributed information processors (13) comprising many simple processing elements. Networks store information about experienced stimuli in the form of connection strengths and network topology and can make that information available. In such a network, interneuron connection strengths are used to encode information, and are modified via a learning strategy. ANNs are characterized by three features: a network topology or architecture, an activation function, and a learning strategy; each is discussed in the following sections. Additionally, an abbreviated history of ANNs is provided and the biological inspiration for the computational model is described.

2.1.1 Biological Inspiration. This is all one big stub.

Fig[an image of a neuron, mapped to a schematic of a neuron, mapped to a processing element]

Integration of magnitude-encoded, rather than frequency-encoded, signals. Threshold functions relationship to the biological shape, size... Papers needed.

Biological neural networks are many orders of magnitude slower than those based in ... It is not the size of the network or the number of interconnections alone which confer upon the human brain its remarkable efficiency (Faggin, 1991). Though size and connectivity are necessary, it is the structure, or topology of the network of interconnections that en

Activation time-line: The concentration of neurotransmitters in the extra-synaptic fluid increases, raising the instantaneous membrane potentiality, possibly breaching the threshold potential causing a fire-and-reset, where there is some physical limit on the rate at which the reset step can occur, thus causing the diminishing response seen in Figure 2 of Neurocomputing, whence the fire causes either excitatory or depressive neurotransmitters to be released at the synaptic clefts formed at the end of the axon, thus propagating the activation through the net.

What is changed in synaptic modification is the efficiency with which a particular post-synaptic site is able to convert neurotransmitters in the surrounding intercellular environment, which is to say in the synaptic cleft, into a change in charge inside the cell.

Activation function: A neuron's activation function is just a phenomenon emerges from the combination of its potential threshold and rest rate, both of which are physiological properties which can change after cell formation. (Instructing Perisomatic Inhibition by Direct Lineage Reprogramming of Neocortical Projection Neurons, 2015)

2.1.2 Historical Overview. The study of ANNs began with a 1943 paper (23) by McCulloch and Pitts. In this paper, McCulloch and Pitts united, for the first time, neurophysiology and formal logic in a model of neural activity. This landmark paper marked the beginning of not only the computational theory of neural networks but also the computational theory of mind, and eventually led to the notion of finite automata (28). In (23) McCulloch and Pitts introduced a very simple model of a neuron, which acted as a threshold-based propositional logic unit. Significantly, McCulloch and Pitts showed that a network of their neuron models, interconnected, could represent a proposition of arbitrarily-

high complexity. Said differently, a network of the neuron models described in (23) can represent any logical proposition. These models are often called McCulloch-Pitts neurons and permit only discrete input values which are summed and compared to a threshold value during a fixed time quantum, and do not possess any learning mechanism. McCulloch-Pitts neurons are able to incorporate inhibitory action, but the action is absolute and inhibits the activation of the neuron without regard to any other considerations. The McCulloch-Pitts neuron model is of theoretical significance, but cannot be applied to practical problems.

Though McCulloch and Pitts made mention of learning in their 1943 paper, thirteen years would pass before the learning concept was formalized into a mathematical and computational model. In 1956 Rochester, Holland et. al. (29) presented the first attempt at using a physiologically-inspired learning rule to update the synaptic weights of a neural network. This model was based on the correlation learning rule postulated in 1949 by Hebb¹. In his book *The Organization of Behavior*, Hebb suggested that synaptic plasticity, that is the capacity of synaptic strengths to change, is driven by metabolic and structural changes in the both neurons near the synaptic cleft (14) such that if two cells often fired simultaneously the efficiency with which they cause one another to fire will increase. This efficiency is now called a synaptic weight. Rochester et. al. showed that the addition of variable synaptic weights alone was not sufficient to produce a network capable of learning; the weights must also be capable of assuming inhibitory values.

[Graphic Stub: simple perceptron terminology, figure.]

The next major contribution to the field would come in 1958 with Rosenblatt's introduction of the simple perceptron (30). The perceptron was the first (2) well-formed, computationally-oriented neural network. Crucially, and unlike most preceding neural models, the model Rosenblatt presented in his 1958 paper was associative. That is, the model learned to associate stimuli with a response. This learning is accomplished by modifying the synaptic weights of the model such that the difference between an input pattern and the desired output pattern is minimized. The responsibility for the error, or difference

¹It should be mentioned that, while Hebb was the first to postulate the correlation learning rule as it relates to neurons and synaptic connection strength, the abstract rule was foreshadowed as early as 1890 by William James (17) in Chapter XVI of *Psychology (Briefer Course)*.

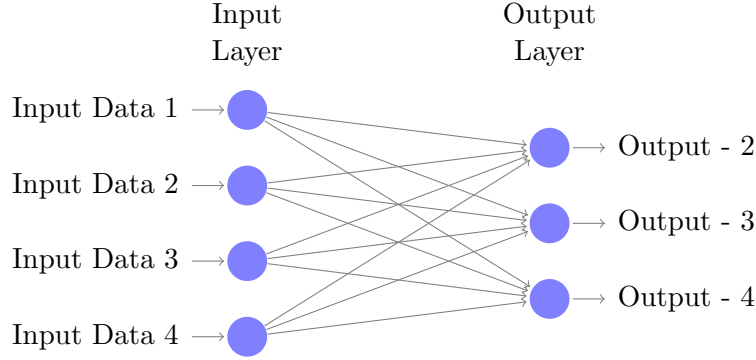


Figure 2.1 A simple perceptron.

between the correct and computed output patterns, is divided among the weights in proportion to their magnitude. Thus, large synaptic weights will be reduced more than small synaptic weights for a large, positive error. This weight update strategy is represented mathematically as:

$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j)x_{j,i} \quad (2.1)$$

where $w_i(t)$ the synaptic weight for feature i at discrete time t , α is the tunable learning rate parameter, d_j is the desired output, y_j is the computed output, and $x_{j,i}$ is the input pattern. This method constitutes a form of reinforcement learning.

Rosenblatt's perceptron was found to be successful at predicting the correct response class for stimuli only if the responses were correlated. It was not until Block's 1962 publication that the reason for this observed performance was elucidated. In this paper, Block presented two key findings: first, that simple perceptrons require linearly separable classes to achieve perfect classification and second, the perceptron convergence theorem (5). Linear separability is the ability of the response classifications to be separated by a hyperplane in the n -dimensional space of the input stimuli to which they correspond. The requirement of linear separability arises directly from the way in which the output of a perceptron response unit is calculated. The output of a perceptron response unit is given by:

$$y_j = \begin{cases} -1 & \sum_{i=1}^n w_{i,j}x_i \leq \Theta \\ +1 & \sum_{i=1}^n w_{i,j}x_i > \Theta \end{cases} \quad (2.2)$$

where y_j is the response value of response unit j , $w_{i,j}$ is the synaptic weight of the connection between activation unit i and response unit j , x_i is the activation value of activation unit i , and Θ is the threshold value of the perceptron. Block's crucial observation was that the form of the summation in the response determining equation is isomorphic to a hyperplane in an n -dimensional space. Thus, in order for the perceptron to achieve perfect classification, a hyperplane must be able to separate them in the n -dimensional input space. The corollary of this observation is the perceptron convergence theorem. The theorem proves that for some learning rules, if a perfect classification is possible it will be found by the perceptron. Specifically, the class of learning rules which were found effective were those which did not change synaptic weights when a correct classification occurs. While the condition does ensure convergence, it often causes very slow convergence, as the synaptic weights change much more slowly when only a small number of samples remain misclassified. Considerably faster guaranteed convergence can be achieved using an error gradient descent learning rule (36), as described by Widrow and Hoff.

In 1969 Minsky and Papert published *Perceptrons*, a book on mathematics and theory of computation. In this book Minsky and Papert mathematically and geometrically analyzed the limitations inherent in the perceptron model of computation. The authors reasoned that each response unit of Rosenblatt's perceptrons was actually computing logical predicates about the inputs it received, based on the observation that response units can either be active or inactive. This analytical framework allowed the authors to construct unprecedented geometric and logical arguments about the computational capabilities of a perceptron. They found that there were several classes of problems which were unsolvable by linear perceptrons (25). In the final chapter of the *Perceptrons* Minsky and Papert extended their judgments regarding the ineffectiveness of single-layered perceptrons to all variants of perceptrons, including the multi-layered variety. This conjecture would turn out to be one of the most significant of the entire book, as it likely resulted in a reduction of funding for neural network research (2) which lasted for several years. Unfortunately, this judgment was incorrect.

While it is true that the pace of development in the field of neural networks slowed considerably after the publication of *Perceptrons*, there was still progress made during the

1970s. In 1972, both Anderson (1) and Kohonen (19) published models of what would come to be known as linear associative neural networks, which are a generalization of Rosenblatt's perceptron. As with the perceptron, neurons in a linear associative neural network compute their output by summing the product of each input signal the synaptic weight associated with that input. Unlike the perceptron, the output of these networks is proportional to this sum, rather than a binary value computed by applying a threshold function to the sum. Though still unable to achieve perfect classification on many classes of problems, these networks were able to successfully associate input patterns with output patterns.

The decade also saw the advent of self-organized maps, which are a type of competitive learning neural network. Self-organization in neural networks was first demonstrated by van der Malsburg in a paper (34) published in 1973. This paper analyzed the response of simulated cortical cells to a simulated visual stimulus. The paper is interesting both for the complexity of the neural model developed, and because it contained the first direct comparison between computer simulation and physiological data (2).

Throughout the decade progress was also made in the understanding of the physiology of biological neural networks. Of particular interest are those papers describing the lateral retinal system of *Limulus polyphemus*, the horseshoe crab. Chosen for the easy with which experiments may be conducted on its compound lateral eye, *Limulus* features prominently in the neurophysiological research. Several works were published on the *Limulus*, perhaps the most significant of which came near the end of the decade with the 1978 publication of a paper describing the dynamics of the retina of a *Limulus* when exposed to moving stimuli. In this paper, the *Limulus* eye was analyzed as a linear system, and the results of this analysis were compared to the actual response of the system to input pattern. The agreement between the linear² model and the biological output signals was found to be in excellent agreement (6). This finding was interesting for the purposes of perceptron simulation, but was ultimately found not to hold for larger collections of neurons.

²Linear, in this context of this system, means that the output of the system when presented with the sum of a set of inputs is equivalent to the sum of the outputs of the system when presented with each input individually.

Several events conspired to create a reinvigoration of neural network research in the early 1980s. Theoretical advances in the physiology of biological neural networks. processor manufacturing technology had advanced sufficiently to allow for much larger-scale simulations. simulation capability...

In 1982, John Hopfield published *Neural networks and physical systems with emergent collective computational abilities*. This momentous work is regarded by many to be the beginning of the renaissance of neural network research (2), and contains many novel insights. Hopfield begins the paper differently than past researchers. Rather than proposing a learning rule or network topology and then evaluating the results of this proposition, Hopfield begins by considering an alternative purpose for a neural network. Hopfield suggests that the network be thought of as a means to develop locally stable points, or attractors, in a state space. The state space comprises the set of states which are the activation value of each neuron. Thus, learning should be the process of modifying the synaptic weights such that they cause the system to flow into local attractors which represent the desired output. In such a model, a noisy or incomplete input would result in an activation pattern which resides on a gradient in the state space. The neural network would then change the activation pattern in such a way as to move the system down the gradient into the attractor state. Hopfield suggests that this process is a general physical description of the concept of content-addressable memory.

[Graphic Stub: notional activation state space with letters and noisy letters as points.]

Hopfield then proposes a network architecture to achieve this behavior (15). The chosen model is one which has binary neural output values and recurrent connections. Neural networks of this types are now called Hopfield networks. Like Rosenblatt's original perceptron model, the neurons used in Hopfield's work had a non-linear, threshold activation function. The network topology was recurrent, with the restriction that no neuron could provide input to itself. Hopfield adopted a variation of Hebb's learning rule to update the synaptic weights.

In Hopfield's network model, the connection strength between two neurons i and j is denoted as T_{ij} , and the activation status of a neuron i is denoted as V_i . T is therefore

the connection matrix of the neural network, with each element representing an individual connection strength and zeros along the diagonal. It is from this organization of the connection strengths that one of the most important insights of this work originates. Hopfield recognized that, in the special case of the model in which $T_{ij} = T_{ji}$, a quantity E could be defined such that

$$E = -\frac{1}{2} \sum_{i \neq j} \sum T_{ij} V_i V_j. \quad (2.3)$$

The change in this quantity as a result of a change in one of the activation values, V_i in the following equation, is then represented as

$$\Delta E = -\Delta V_i \sum_{j \neq i} T_{ij} V_j. \quad (2.4)$$

From this equation, it is clear that any change in V_i will reduce the value of E . This decrease in E must necessarily continue until some local minimum of the value of E is reached³. Here, Hopfield observed that this case is isomorphic with an Ising model,” referencing the statistical mechanical model of magnetic spins. In this isomorphism, the quantity E maps to the energy of a physical system described by an Ising model. It is difficult to overstate the importance of this observation. It both provided a novel mechanism by which physical theory could be applied to neural networks, and legitimized the study of neural networks as a physical system, encouraging many physicist to join in the development of the theory.

Hopfield constructed a model of the system described in the paper, and presented it with random input patterns⁴. He found that the network can indeed recall a small number of patterns, on the order of approximately 15 percent of the network dimensionality, before the recall error becomes significant.

In 1985, Ackley et. al. extended the neural network model proposed by Hopfield⁵. Hopfield networks are deterministic with respect to energy; by definition any change in a Hopfield network always reduces the energy of the system or leaves it the same. This is a

³An identical conclusion would be reached if V_j was changed instead of V_i . It is merely a matter of convention.

⁴Hopfield calls these input patterns entities or *Gestalts*.

⁵Though a Hopfield network was used for the work done by Ackley, Hinton, and Sejnowski it is not necessary to use network with recurrent connections.

useful property if it is acceptable to find one of many local minima, or attractors. However, if a single, global minima in the state space is sought this model is likely to converge prematurely to a local attractor state. In order to surmount this limitation Ackley et. al. modified the Hopfield neural model to activate stochastically. The probability of state transition, p , is given by

$$p = \frac{1}{1 + e^{-\Delta E/T}} \quad (2.5)$$

where ΔE is the change in energy of the system resulting from a transition to a new state and T is the artificial temperature of the system. Thus the relative probability, P_α/P_β of moving to either of two arbitrary global states, α and β , is defined as

$$\frac{P_\alpha}{P_\beta} = e^{-(E_\alpha - E_\beta)/T} \quad (2.6)$$

which is a form isomorphic to the Boltzmann distribution. Thus, a neural network with transition probabilities described by equation 2.5 is called a Boltzmann machine. The effect of probabilistic state transitions of this form is that state transitions from low energy states to higher energy states are possible, thereby allowing the system to escape local minima.

Inspection of equations 2.5 and 2.6 reveals that the probability of transition is determined by both ΔE and T . A large value of ΔE , which corresponds to a large increase in total energy, will decrease the probability of transition. Conversely, a large value of T will increase the probability of transition for any arbitrary value of ΔE . The systems artificial temperature therefore acts as a tuning mechanism for the exploration of the state space. A high temperature value will result greater exploration of the space state, but will result in less gradient descent and therefore may cause the system to depart the attractor basin of the global minimum. A low artificial temperature parameter may cause premature convergence. Ackley et. al. solved this tuning problem by recognizing a deep connection to another concept born of statistical mechanics: simulated annealing. Simulated annealing decreases the artificial temperature of a system slowly over the course of a simulation, and as a result increases the likelihood that the final state of the system will be the ground state, which is to say the global minimum. This algorithm is discussed in detail in section 2.2.

With a procedure for finding the global minimum of a space state in place, the authors proceeded to construct state spaces for which the global minimum was of interest. One way to construct such a state space is to include hidden units in the neural network. Hidden units are neural units which are neither input nor output units. These hidden units allow the network to solve interesting problems that are out of reach of simple associative neural networks(2). However, like all neurons in any neural network, the connection weights of these neurons must be modified in order for the network to learn. It is not immediately clear how hidden unit synaptic weights can be modified to account for the performance of the network. This deficiency is often called the credit assignment problem(4). The application of simulated annealing in Boltzmann machines avoids the problem of assigning credit to hidden units, thereby enabling their inclusion in the model. This is historically significant because it was the first successfully-implemented multi-layered neural network (13).

The next major development in neural networks came in 1986 with the introduction of the back-propagation algorithm. Though the formalisms required involved in this algorithm had been developed earlier (7, 35), and the method was simultaneously discovered independently by two other groups (27, 21), it was Rumelhart et. al. that applied the algorithm to machine learning (31). Back-propagation can be thought of as a generalization of the gradient descent⁶ algorithm presented by Widrow and Hoff (36), which includes the errors associated with connection strength of hidden units, or internal representation units. A detailed discussion and derivation of the back-propagation algorithm is included in Section 2.1.5.1. Though back-propagation in multilayered perceptrons cannot be guaranteed to find an exactly correct solution, the algorithm is demonstrably capable of solving difficult and interesting problems, thus disproving the speculation of Minsky and Papert in (25).

With the advent of Boltzmann machines and back-propagation, it became possible to analyze the properties and capabilities of multilayered neural networks. In 1989 Cybenko showed that a multilayer feedforward neural network with nonlinear activation function is

⁶The gradient descended in this context is the gradient of the error surface in the space of states of synaptic weights, not the gradient of the activation state surface, as with a Hopfield network.

in principle capable of approximating any continuous function (9). This finding is striking because it implies that, given the correct learning rule and a sufficiently large network, a multilayer neural network can learn a pattern of arbitrary complexity.

2.1.3 Network Topology. Stub.

2.1.4 Activation Functions. Stub.

2.1.5 Learning Strategies. Stub.

2.1.5.1 Back Propagation Training. Stub.

2.1.5.2 Simulated Annealing Training. The first application of simulated annealing (SA) to the training of neural networks was accomplished by Engle in (11), with limited success. Engle's work involved discrete

2.2 Related Works in Simulated Annealing

Simulated annealing is a stochastic optimization algorithm which can be used to find the global minimum of a cost function mapped from the configurations of a combinatorial optimization problem. The concept of simulated annealing was introduced in by Kirkpatrick et al. in (18) as an application of the methods of statistical mechanics to the problem of discrete combinatorial optimization. Specifically, simulated annealing is an extension of the Metropolis-Hastings (24) algorithm which can be used to estimate the ground energy state of a many-body systems at thermal equilibrium. Kirkpatrick et al. applied the Metropolis-Hastings algorithm sequentially, with decreasing temperature values in order to approximate a solid slowly cooling to low temperatures. Later work by Goffe (12), Corana et al. (8), and Lecchini-Visintini et. al. (20) extended SA to the continuous domain.

In the most general terms, SA is a local search algorithm through the problems solution space, \mathcal{S} , which is the set of all possible solutions, \mathbf{s} , of the problem. The search is conducted by generating new solutions to the problem by applying a neighborhood

function, \mathcal{N} to the current solution; the neighborhood function specifies the way in which a solution is transformed to yield a new solution, or neighbor solution, and is generally problem dependent. To apply SA to an optimization problem it must be possible to characterize each possible solution of the problem using a cost function, \mathcal{C} , where \mathcal{C} is the mapping

$$\mathcal{C} : \mathcal{S} \rightarrow \mathbb{R}.$$

Because \mathcal{C} is a function on \mathcal{S} , it is said that the cost function forms a cost surface in the solution space. During each iteration of the algorithm, a new solution, \mathbf{s}' , is generated using $\mathcal{N}(\mathbf{s})$, and the cost of that solution, $\mathcal{C}(\mathbf{s}')$, is determined. The change in cost associated with moving from the current solution to the neighbor solution is given by

$$\Delta C = \mathcal{C}(\mathbf{s}') - \mathcal{C}(\mathbf{s}).$$

ΔC is then used in conjunction with an artificial temperature parameter to determine if the newly-generated neighbor solution is to become the current solution. The artificial temperature parameter is specified by the temperature schedule, $T(t)$, where t is the number of simulation iterations completed. The temperature controls the probability of the system moving to a higher cost solution, thereby enabling the algorithm to escape local minima on the cost surface. In the parlance of SA (18) a system at its maximum temperature is said to be *melted*. In the melted state, most neighbor solutions are accepted by the algorithm. Analogously, a system that has a temperature of zero, which indicates that the algorithm cannot move to any higher-error state, is said to be *frozen*. Note that a frozen system may still be perturbed into a lower-energy state. The notions of freezing and melting enter the SA algorithm in the form of the acceptance criterion, which determines if a newly-generated solution is to become the current solution. The most commonly used acceptance criterion is the Metropolis criterion (24), given by:

$$y_j = \begin{cases} 1 & \Delta C \leq 0 \\ e^{-\frac{\Delta C}{T(t)}} & \Delta C > 0 \end{cases} \quad (2.7)$$

The probability of moving to a solution which is higher in cost than the current solution is derived from the statistical mechanical probability of traversing a potential energy barrier by thermal fluctuations. When considering only the influence of classical thermal fluctuations in particle energy levels, the probability of a particle traversing a barrier of height ΔV at a temperature T is on the order of

$$\mathcal{P}_t = e^{-\frac{\Delta V}{T}}. \quad (2.8)$$

The SA algorithm is presented in Algorithm 1.

Algorithm 1 Simulated Annealing

```

 $s \leftarrow s_0$ 
 $t \leftarrow 0$ 
while  $T(t) > \epsilon$  do
   $s' \leftarrow \mathcal{N}(s)$ 
   $\Delta C \leftarrow (\mathcal{C}(s') - \mathcal{C}(s))$ 
  if  $\Delta C \leq 0$  then
     $s \leftarrow s'$ 
  else  $\{\exp(\Delta C/T(t)) > U(0, 1)\}$ 
     $s \leftarrow s'$ 
  end if
   $t \leftarrow t + 1$ 
end while
 $s_{opt} \leftarrow s$ 
return  $(s_{opt})$ 

```

In (32) Szu and Hartley introduced the method of fast simulated annealing (FSA), which incorporates occasional, long jumps through the configuration space. These jumps are accomplished by using a heavy-tailed distribution, such as the Cauchy distribution, for the visiting distribution used in the neighborhood function. This provision increases the likelihood of escaping local minima, and reduces the total computational effort required to reach a global minimum. This modification yields a significant decrease in the amount of computation effort required to guarantee that a global minimum is found. Specifically,

the FSA decreased the required temperature decay from $1/\ln(t)$ to $1/t$, where t is the simulation time.

Later work by Tsallis and Stariolo (33), generalized both CSA and FSA into a single framework: generalized simulated annealing (GSA).

2.2.1 Reannealing. In (16), Ingber introduced the concept of reannealing, which is a mechanism which allows for the artificial temperature parameter to be occasionally increased. The increase, or rescaling, of the temperature parameter enables the SA algorithm to move to higher cost solutions, effectively restarting the annealing process, but from a configuration space location that is already known to be a local minima. This mechanism allows the SA algorithm to escape from local minima, and thus decreases the simulation time required to avoid final convergence to a global minima.

2.3 Related Works in Quantum Mechanics

Quantum mechanics is the branch of physics concerned with the physical laws of nature at very small scales. Many aspects of physical reality are observable only at these scales. Several techniques described in this document are either inspired by, or are simple models of quantum mechanical processes. These concepts are very briefly reviewed in this section.

2.3.1 Quantum Tunneling. One of the quantum phenomena for which there is no classical analog is potential barrier penetration, also known as quantum tunneling. This phenomenon arises from the probabilistic and wavelike behavior of particles in quantum physics. Tunneling plays a significant role in the behavior of bound and scattering quantum mechanical systems.

A particle with energy E incident upon a potential energy barrier of height $\Delta V > E$ has a non-zero probability of being found in, or past, the barrier. Classically, this behavior is forbidden. The probability of tunneling, \mathcal{P}_t , through a step barrier of height ΔV is described by:

$$\mathcal{P}_t = e^{-\frac{w\sqrt{\Delta V}}{\Gamma}} \quad (2.9)$$

where Γ is the tunneling field strength (26). Fig. 2.2 depicts a one-dimensional example of the quantum tunneling of the probability distribution function of the location of a particle indecent upon a potential energy barrier.

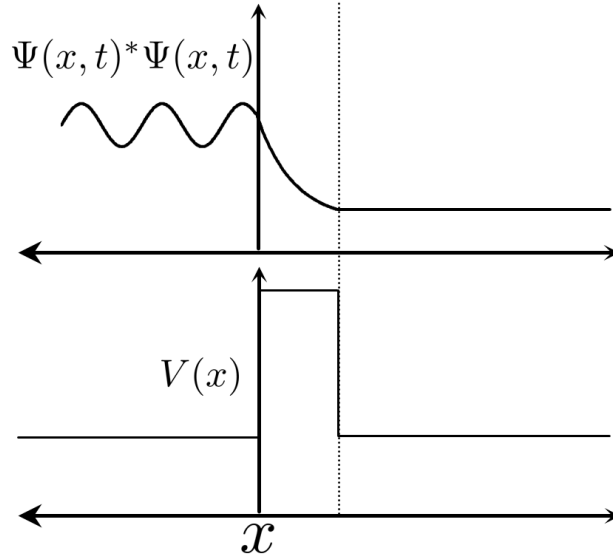


Figure 2.2 (Bottom) A simple step potential in one dimension. (Top) The probability density function of a generic quantum system in the presence of a potential energy step barrier. There is an exponential decrease in probability through the barrier, and a uniform probability beyond the barrier.

2.4 Notation and Terminology Conventions

There is a great deal of academic writing describing quantum annealing in the language of physics, but very little writing describing the concept from an algorithmic perspective. For this reason a new, more specific term is introduced in this document. Simulated quantum annealing (SQA) is the quantum mechanical counterpart of simulated thermal annealing.

The term neuron will be used in this document to describe the information processing elements of a neural network. This convention is selected both for conciseness and for the useful adjectival form, neural, which will be of great explanatory utility in the coming chapters.

III. Methodology

The application of SA to feed-forward neural network weight selection requires the specification of several formalisms and representations linking the two concepts. In this chapter the representations adopted in this thesis are discussed. These representations will be combined into a set of formalisms which describe the various approaches to the application of SA to feed-forward neural network weight selection. A design of experiments, which will be used to evaluate the performance of these approaches, is established.

3.1 Simulated Annealing

In this thesis, several variations the SA algorithm are developed and implemented. Each is applied to the problem of selecting synaptic weights for a feed-forward neural network in order to maximize the performance of the network; exactly what is meant by performance of the network is discussed in section 3.5. This section contains an abstract discussion of the SA formulations to be applied to this task; later sections will expound the implementation details specific to the feed-forward neural network application of these SA formulations. For all SA formulations examined in this thesis the Metropolis acceptance criterion will be used. The temperature schedule will be determined by the convergence properties of the constructed algorithm. Examining Alg. 1 reveals that the preceding specifications leave only one component unspecified: the neighborhood function, which determines how new trial solutions are generated from the current solution. In the following sections the neighborhood function is decomposed into two decoupled components, the visiting distribution and anisotropy policy, and several possible realizations of each are discussed.

3.1.1 Neighborhood Functions: Traversing the Cost Surface. The SA algorithm requires that a neighborhood function, \mathcal{N} , be specified to produce new solutions from a given solution. The neighborhood function performs the exploration of the solution space, as it specifies new solutions which can be either accepted or rejected according to the acceptance criteria. Thus, \mathcal{N} determines how the algorithm traverses the cost surface of the problem. A traversal action, or move, on a surface, may be decomposed into two

components: the distance moved and the direction of the move. These components may be specified independently of one another. The distance of the move on the cost surface has been examined in previous work (32, 33, 16), and is often specified using a *visiting distribution*, which is defined as a probability distribution of transition to a solution over the solution space of the problem. The visiting distribution specifies the magnitude and the direction of the move

In previous work (33), the move distance has been applied isotropically in all possible dimensions of travel. In physical science, isotropicity is phenomenological property of being uniformly applicable in all dimensions. In the context of neighborhood functions this means that the probability distribution over the solution space is symmetric; that is, that the probability distribution in each dimension is the same. In the following sections a method is developed for specifying an anisotropic visiting distribution.

3.1.1.1 Visiting Distributions. The visiting distribution of a neighborhood function is probability density function over all possible solution states, which gives the probability of transitioning from the current state of the system to each other state. Once a visiting distribution is specified, samples can be drawn from the distribution using inverse transform sampling. These samples may be either drawn from an n -dimensional distribution, where n is the number of free parameters of the problem, or n samples can be drawn and applied to each of the free parameters independently. In this thesis, all visiting distributions are implemented using the latter method. In the following sections several visiting distributions will be discussed.

Gaussian Visiting Distributions. A Gaussian visiting distribution is commonly used in SA. Because it is a light-tailed distribution, and therefore indicates very low probability for all values far from the mean, Gaussian visiting distribution results in a search which is highly local about the mean. The mean of the visiting distribution is always the current solution location in solution space of the SA algorithm. An SA algorithm using a Gaussian visiting distribution is often called classical simulated annealing (CSA) (33), both because it is the formulation of SA that was described first, and because the dynamics of the algorithm are isomorphic with those of classical thermodynamics. In this work, a

standard normal distribution is used. The probability density function for this distribution is given by

$$g_G(x) = \frac{e^{-\frac{1}{2}x^2}}{\sqrt{2\pi}}. \quad (3.1)$$

Cauchy Visiting Distributions. Local search must occur in order to enable gradient descent, but it introduces a limitation. If the artificial temperature, which controls the probability of moving uphill on the cost surface, is lowered too quickly the algorithm can be caught in a local, rather than global, minima; this is also known as the freezing problem. One way to alleviate this limitation is to construct a neighborhood function which enables the system to escape local minima by means other than hill-climbing. In quantum mechanics, a system which is trapped in a local minimum on a potential energy surface may escape that minima by tunneling through the potential energy surface to a lower energy state. It is possible to construct several visiting distributions which act analogously to quantum tunneling. This can be done by using a visiting distribution that has a non-negligible probability of generating large traversal distances. If the traversal distances generated are sufficiently large, it is possible that the arrived-at solution will be across the cost surface barrier surrounding the local minima, thus allowing the algorithm to escape the minima. The term *quantum-inspired visiting distribution* will be used in this thesis to describe any distribution possessing this property.

The significant performance gains exhibited by the FSA algorithm are the result of using a quantum-inspired visiting distribution. In (32), which introduces the FSA algorithm, a Cauchy distribution is used to generate new solutions. Unlike the Gaussian distribution, the Cauchy distribution is heavy-tailed, meaning that it will occasionally produce values which are relatively far from the mean. This property of the distribution has the useful consequence of increasing the probability of escaping a local minima by allowing the algorithm to tunnel through the cost-surface barriers that surround it. This in turn allows for faster convergence relative to CSA. To understand the origin of this advantage, it is instructive to contrast equations 2.8 and 2.9. Both describe the same value, but the importance of the width and height of the traversed barrier in the two equations is considerably different. For systems in which quantum tunneling is possible, the probability

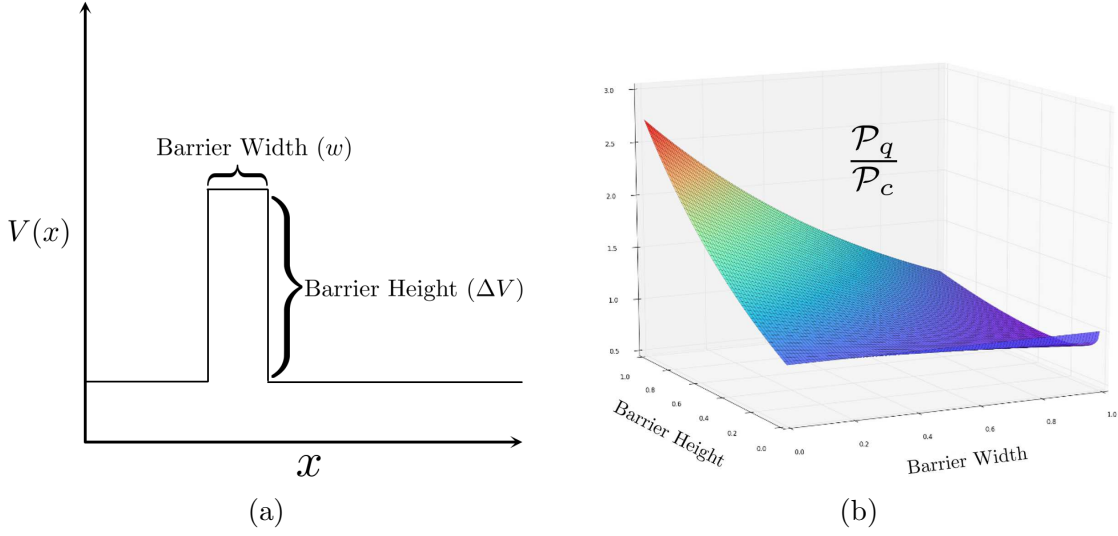


Figure 3.1 (a) A potential energy barrier on a one-dimensional potential energy surface. (b) The tunneling probability relative to the probability of traversal due to thermal fluctuation for a step barrier plotted as a function of the height and width of the barrier.

of penetrating a barrier of height ΔV is increased by a factor of approximately $e^{\Delta V}$, for large values of ΔV . This relationship is depicted graphically in Fig. 3.1 which shows the probability of barrier traversal for a system which allows quantum fluctuations, divided by the same probability for a system which only considers thermal fluctuations. Fig. 3.1 illustrates the fact that physical models which considers quantum effects are much more likely to predict penetration of tall, thin energy barriers than those which only include classical thermal effects.

The general probability density function for the Cauchy distribution k is given by

$$g_C(x) = \frac{1}{\pi\gamma} \frac{\gamma^2}{(x - x_0)^2 + \gamma^2}$$

where x_0 is the mean and γ is a shape parameter. In this work, γ will always be set to 1 and the mean will always be 0, yielding the specific Cauchy distribution given by

$$g_C(x) = \frac{1}{\pi x^2 + \pi}. \quad (3.2)$$

sophisticated form of SA is GSA, described by Tsallis and Stariolo in (33). As the name implies, this SA implementation is a generalization of other forms of SA, specifically CSA and FSA, which can be recovered under certain conditions in the GSA formulation. As with FSA, GSA is essentially SA with a modified visiting distribution. The visiting distribution proposed in (33) is given by

$$g_{GSA}(x) = \left[\left(\frac{q_V - 1}{\pi} \right)^{(D/2)} \right] \left[\frac{\Gamma \left(\frac{1}{q_V - 1} + \frac{D-1}{2} \right)}{\Gamma \left(\frac{1}{q_V - 1} - \frac{1}{2} \right)} \right] \left[\frac{T_{q_V}^{-\frac{D}{3-q_V}}}{\left(1 + \frac{(q_V - 1)(x^2)}{T_{q_V}^{2(3-q_V)}} \right)^{\frac{1}{q_V - 1} + \frac{D-1}{2}}} \right]. \quad (3.3)$$

where D is the dimensionality of the solution space to be searched, q_V is a free parameter which is selected by the experimenter, and T_{q_V} is an introduced stochastic process control parameter¹ which may, or may not, change during the execution of the SA algorithm. Unlike the Cauchy and Gaussian visiting distributions used in CSA and FSA, the GSA visiting distribution has several free parameters which alter the shape and scale of the distribution. The GSA visiting distribution should therefore be conceptualized as a family of related distributions, from which one may be selected to construct a GSA neighborhood function.

Fig. 3.2 contains a comparison between the distributions produced by several variations of q_V and T_{q_V} , with D set to 1. Values of q_V which are near 1 yield very light-tailed distributions, which are similar to Gaussian distributions. As $q_V \rightarrow 1$, $g_{GSA}(x)$ recovers $g_G(x)$. Similarly, as $q_V \rightarrow 2$, $g_{GSA}(x)$ recovers the Cauchy distribution, $g_C(x)$; this behavior is seen in the $q_V = 2$, $T_{q_V} = 1$ case displayed Fig. 3.2, in which the GSA distribution exactly recovers the Cauchy distribution. Values of q_V greater than 2 do not have a inde-

¹This temperature parameter is completely independent from the annealing temperature parameter of SA. The two parameters can vary independently of on another, but will both be annealed over the course of the algorithm.

pendent analog distribution and have been experimentally shown [bunch of cites here] to yield more efficient SA algorithms, when used as a visiting distribution.

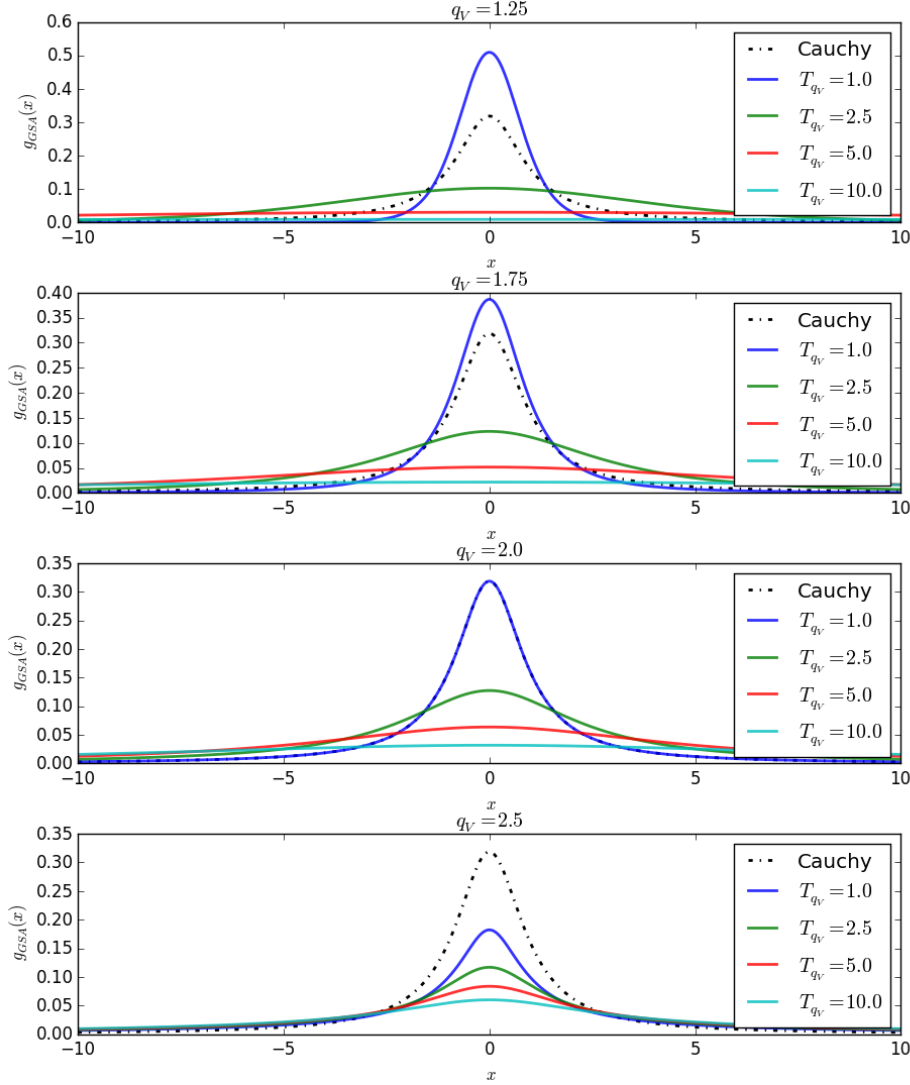


Figure 3.2 This figure comprises four plots, each of which displays a GSA distribution at various values of T_{q_V} , in order to illustrate the behavior of the GSA distribution for several values of q_V . This figure shows the GSA near the mean, which illustrates the effect of q_V and T_{q_V} on the near-mean domain values, while neglecting the effects on the tails of the distribution.

Several trends can be ascertained through the examination of figures 3.2 and 3.3. The former displays the behavior the GSA distribution near the origin, where, for most combinations of q_V and T_{q_V} , the majority of the probability mass is concentrated; the

latter details the behavior of the GSA distribution at domain values far from the origin, or, in the tails of the distribution. As is shown in Fig. 3.2, q_V primarily influences the spread of the probability mass, or shape, of the distribution. Values of q_V near 1 produce distributions which have only negligible probabilities of generating values far from the mean, while larger values increase in statistical dispersion. This behavior is particularly clear in Fig. 3.3, which shows the distribution over a large portion of the domain and very near the origin of the range. Examining the figure from top to bottom, in order of decreasing q_V , it is clear that increasing q_V corresponds to an increase in the probability of a sample yielding a value far from the mean. A larger q_V value also corresponds to less tail-behavior sensitivity to the temperature parameter, T_{q_V} . T_{q_V} also influences the distribution shape. As the value of T_{q_V} is increased, the distribution becomes more uniform over the domain. This has important consequences for the application of GSA distribution to stochastic search problems such as SA.

As discussed in (10, 3) the distributions produced by Eq. 3.3 have several useful properties, when used in stochastic search procedures. The longer tails of the GSA distribution when $q_V > 1$ enable more homogeneous visitation of the entire solution space of the problem, relative to a Cauchy distribution. Furthermore, the fact that q_V is selected by the experimenter creates an opportunity for problem-specific construction of the visiting distribution used in the SA implementation. The newly-introduced temperature parameter, T_{q_V} , may also be exploited to escape cost surface local minima. Regardless of the value of q_V , large values of T_{q_V} produce distributions which have high statistical dispersion, and are therefore able to produce problem configurations which far, in configuration space, from the current state.

While this distribution produces a neighborhood function with several advantageous search characteristics, it does have a significant drawback: The integral of Eq.3.3 has no closed-form analytic solution. The indefinite integral, but this operation yields the hypergeometric function, [function goes here], which can only be computed as a power series of increasingly complex terms. Previous work (10) has shown that this procedure is computationally expensive. Thus, Eq. 3.3 is unsuitable for the analytic inverse sampling transform method of random variable generation. In order to overcome this limitation, a

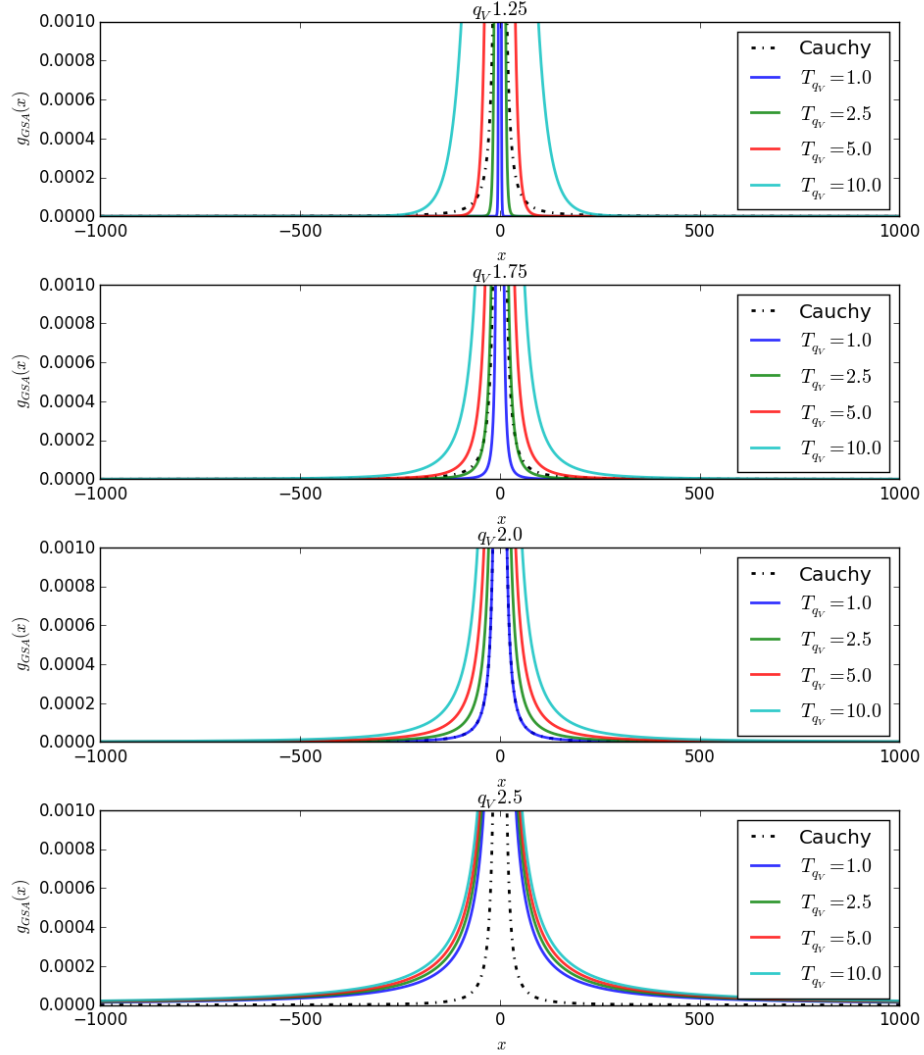


Figure 3.3 This figure comprises four plots, each of which displays a GSA distribution at various values of T_{q_V} , in order to illustrate the behavior of the GSA distribution for several values of q_V . This figure shows the GSA distribution for domain values far from the mean, which illustrates the effect of q_v and T_{q_V} on the tails of the distribution.

distribution sample caching method was implemented. This method works by numerically approximating the integral of the distribution given by Eq. 3.3, which yields a series of domain-range value pairs which constitute a close approximation to the cumulative distribution function of the distribution. This series of values, and the displacement values to which they correspond are stored. To sample the distribution, a number is selected

with uniform probability from the interval $[0, 1]$. The numeric integration value which has the minimum-magnitude difference with the randomly selected value is identified, and the displacement value to which it corresponds is returned. This procedure is the numerical equivalent of inverse transform sampling.

An arbitrarily-large set of samples can be constructed using this method; If the sample set is sufficiently large, a random choice from the sample set is approximately equivalent to sampling the original distribution. A large set of samples for each combination of q_V , T_{q_v} , and D can be stored for later access, thus enabling fast numerically-approximate sampling of Eq. 3.3. A system fitting this description was constructed in support of this thesis. A thorough search of the publicly-available resources indicates that there are few systems available for the generation of GSA random variables, despite the popularity and utility of GSA. As such, the GSA sample generation system constructed in support of this thesis will be made publicly-available upon publication of this thesis.

Uniform Visiting Distributions. As illustrated in Fig. 2.2 the probability of observing a particle beyond a classically-impenetrable potential barrier is uniform beyond the barrier². An analogous visiting distribution can be constructed, which models all configuration space movement distances as equally likely. The utility of this visiting distribution is that it makes the entire cost surface accessible each time the neighborhood function is applied to the current state; this property is useful, but prevents any local gradient descent. As such, the uniform visiting distribution serves as a useful upper bound in the trade-space between global and local search policies.

3.1.1.2 Anisotropy Policies. In previous work, the visiting distribution is applied isotropically over the free parameters, or dimensions, of the solution space (33, 10). In the context of SA, isotropic application of the visiting distribution means the next state for each free parameter is a sample from a common, identical distribution.

²This observation only holds for potential energy surfaces containing a single barrier. The analogous cost surface over neural network weight space is likely to have many barriers corresponding to the superimposed convex spaces around competing conventions of weight configurations which encode similar functions. Thus the analogy used here is only an approximation of the behavior of quantum systems.

Isotropic Anisotropy.

Adjacency-Based Anisotropy.

Neural networks may be viewed as a collection of interacting subgraphs. Each subgraph in a complete feed-forward neural network produces a

State-Based Anisotropy.

When evaluating the traversal characteristics of the synaptic annealing algorithms constructed in this thesis, it was observed that the sum of squared weight values grew considerably during training. Previous work () has shown that large weight magnitudes result in networks with high bias, and correspondingly high generalization error. It is intuitively clear that, when using a sigmoidal activation function, a very high-magnitude weight value effectively converts the afferent neuron, relative to that synapse, into a bias. Unless an equally large-magnitude weight of opposite sign is introduced, or the weight value fluctuates to a smaller value, the afferent neuron will remain a bias. One of these contingencies may occur, but it is unlikely that such an event would reduce the total training error of the network, and is therefore likely to be rejected. In (22), Lee et al. propose a method called multiobjective hybrid greedy simulated annealing (MOHGSA), which uses SA to minimize both the cost function and the sum of squared weights. By doing so, Lee et al. were able to reduce the generalization error of a neural network trained by SA. The MOHGSA approach works by exerting a selection pressure which lowers the likelihood of the SA algorithm accepting moves which reduce the cost function if they increase the sum of squared weights.

3.2 Feed-Forward Neural Network Representation

The problem of feed-forward neural network synaptic weight selection must be formulated as a combinatorial optimization problem before any formulation of SA can be applied to it. Each synaptic weight in a feed-forward neural network may be encoded as a real-valued element in a 3-dimensional relation matrix, denoted as W_{ijk} . In this encoding scheme, for a given layer, k , of the matrix the row and column indexes indicate the presynaptic and post-synaptic neurons, respectively. The absence of a synaptic connection is indicated by a value of 0 in the matrix element corresponding to that synaptic connec-

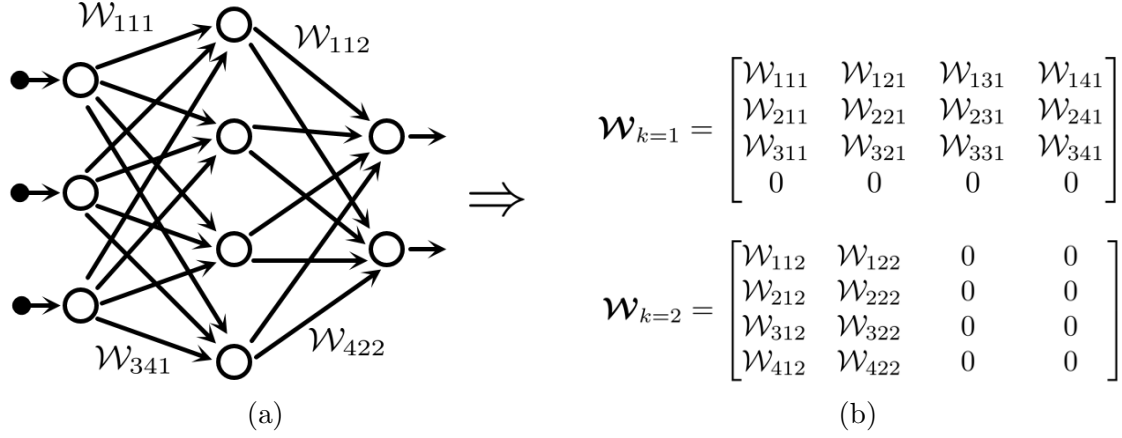


Figure 3.4 (a) An arbitrary feed-forward ANN. (b) The weight matrix representation of the feed-forward ANN in (a).

tion. A nonexistent synapse can be caused by the absence of either the presynaptic or post-synaptic neuron, or by the absence of a connection between the neurons. This weight encoding scheme is depicted graphically in Fig. 3.4. The weight matrix, \mathbf{W} , therefore encodes a configuration in the solution space of the problem, and can be visualized as:

$$\mathbf{W} = \begin{bmatrix} \begin{bmatrix} w_{111} & w_{121} & \dots & w_{1j1} \\ w_{211} & w_{221} & \dots & w_{2j1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{i11} & w_{i21} & \dots & w_{ij1} \end{bmatrix} & \begin{bmatrix} w_{112} & w_{122} & \dots & w_{1j2} \\ w_{212} & w_{222} & \dots & w_{2j2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{i12} & w_{i22} & \dots & w_{ij2} \end{bmatrix} & \dots & \begin{bmatrix} w_{1jk} & w_{12k} & \dots & w_{1jk} \\ w_{2jk} & w_{22k} & \dots & w_{2jk} \\ \vdots & \vdots & \ddots & \vdots \\ w_{ijk} & w_{i2k} & \dots & w_{ijk} \end{bmatrix} \end{bmatrix}$$

The total solution space, \mathcal{S} , may then be defined as the set of all possible configurations of \mathbf{W} for a given neural network. \mathbf{W} is, in effect, the phase space of the feed-forward neural network system. In the synaptic weight selection problem domain it is more evocative to call \mathcal{S} the weight space of the network, so this convention adopted throughout this thesis.

A network representation has now been specified, but a formal description of the data on which the network is to operate remains to be constructed. Feed forward neural networks associate a set of numeric input data elements with a set of numeric output data elements. Thus the formalism which is used to represent data should capture this directional association. In this thesis, a set of ordered pairs is used to represent this

association. Formally, a set \mathcal{X} is defined such that each element of \mathcal{X} is an ordered pair of the form

$$\{\chi, \{\chi, \lambda\}\}$$

where χ and λ are independent sets of real numbers. In this formalism χ is the input data, and λ is the desired output data. \mathcal{X} is therefore a specific set of ordered pairs mapping input data to output data. Extending this notion, the notation \mathcal{X} is introduced to represent the infinite set of all possible realizations of \mathcal{X} .

In the interest of concise notation, a function $\Phi(\mathcal{W}, \chi)$, which represents set of numeric data produced by propagating the set χ through a feed forward neural network with weights specified by \mathcal{W} , is introduced. Unless otherwise specified, all neural networks discussed in this thesis use a hyperbolic tangent activation function.

Given the formalism \mathcal{S} , for the weight space, and \mathcal{X} for the set of all possible data sets, we define a cost function \mathcal{C} to be the mapping

$$\mathcal{C} : \mathcal{S}, \mathcal{X} \rightarrow \mathbb{R}.$$

For every data set \mathcal{X} , each possible synaptic weight configuration, \mathcal{W} , then corresponds to some cost value $\mathcal{C}(\mathcal{W}, \mathcal{X})$. Thus $\mathcal{C}(\mathcal{W}, \mathcal{X})$ defines a cost surface embedded in the weight space. The objective is now to find a synaptic weight configuration, \mathcal{W}_{opt} such that

$$\mathcal{C}(\mathcal{W}_{opt}, \mathcal{X}) \leq \mathcal{C}(\mathcal{W}, \mathcal{X}), \forall (\mathcal{W} \in \mathcal{S}).$$

With this framework in place, the SA can be applied to transition from a randomly-selected initial state, \mathcal{W}_0 , to \mathcal{W}_{opt} . In this thesis all weights were initialized to a value drawn from a uniform distribution over the range $[-0.1, 0.1]$.

3.3 Applying Simulated Annealing to Feed-Forward Neural Network Weight Modification

The formulations of SA described in Sec. 3.1 may be applied to any properly formulated combinatorial optimization problem. The representation formalism for the weight parameters of a feed forward neural network presented in Sec. 3.2 can serve as the parameter space in a combinatorial optimization function, and thus enables the application of SA to the problem of neural network weight selection. With a more concrete problem definition in place, it is now possible to precisely specify several SA neighborhood functions for the problem of neural network weight selection. In the following sections, several complete definitions of neighborhood functions are presented for the application of SA to feed-forward neural networks. The term *synaptic annealing* is introduced to represent any artificial neural network training algorithm which modifies synaptic weights using SA.

3.3.1 Synaptic Annealing Neighborhood Functions. A generic synaptic annealing neighborhood function may be defined as

$$\mathcal{N}(\mathcal{W}) = \mathcal{W} + \alpha(\mathcal{G} \circ \mathcal{A}) \quad (3.4)$$

where α is the learn rate parameter, \mathcal{G} is the neighborhood sample matrix, and \mathcal{A} is the anisotropy matrix.³ Each element in the neighborhood sample matrix, \mathcal{G} , is a sample generated using the random variable generation function $G^{-1}(U)$, where G^{-1} is the sample generation function for a visiting distribution $g(x)$, and U is a uniform random variable over the range $[0, 1]$. Generally, G^{-1} is the inverse transform of the cumulative distribution function of the visiting distribution, or a numeric approximation thereto. Both \mathcal{G} and \mathcal{A} have dimensionality equal to that of \mathcal{W} . Additionally, the following constraint is imposed on the matrices:

$$\forall \mathcal{W}_{i,j,k} \in \mathcal{W}, [\mathcal{W}_{i,j,k} = 0] \rightarrow [\mathcal{A}_{i,j,k} = 0] \wedge [\mathcal{G}_{i,j,k} = 0]. \quad (3.5)$$

³The symbol \circ denotes the Hadamard product operation, which is simply the element-wise product of two matrices.

The constraint specified in Eq. 3.5 ensures that synaptic connections which are not specified to exist, and are therefore set to exactly 0, are not inadvertently created by the neighborhood function.⁴ The term $\alpha(\mathbf{G} \circ \mathbf{A})$ in Eq. 3.4 is added to the current weight matrix to produce a new weight matrix, and can therefore be thought of as a perturbation of the current state. In the following sections several realizations of this general form, each corresponding to a different form of SA, are presented.

3.3.1.1 Gaussian (CSA) Synaptic Annealing Neighborhood. The original description of SA employed a Gaussian visiting distribution (18). The Gaussian neighborhood function for feed-forward neural networks, \mathcal{N}_g , is defined as

$$\mathcal{N}_G(\mathbf{W}) = \mathbf{W} + \alpha(\mathbf{G}_G \circ \mathbf{A}) \quad (3.6)$$

where α is the learning rate, \mathbf{G}_g a matrix of samples drawn from the standard normal distribution, and \mathbf{A} is an anisotropy matrix.

The neighborhood function given in Eq. 3.6 is an application of the canonical form of simulated annealing to the problem of selecting a weight configuration for a feed-forward neural network. The term classical is used here because the underlying simulated annealing model can be described entirely in terms of classical statistical mechanics. To interpret this in terms of the analogy present in Sec. 3.1.1.1, the probability of the Gaussian visiting distribution used in CSA generating a weight space distance large enough to transition the system across a large energy barrier is effectively 0. In the following sections, models which approximate quantum mechanical phenomena will be constructed.

3.3.1.2 Cauchy (FSA) Synaptic Annealing Neighborhood.

3.3.1.3 Tsallis (GSA) Synaptic Annealing Neighborhood. ... The generation of samples from this distribution is troublesome because it involves the inversion of a

⁴Strictly speaking, it is possible for a weight to be set to exactly 0 in the normal operation of the SA algorithm. If this were to occur, the constraint specified in Eq. 3.5 would prevent that weight from ever being modified again. This scenario was never observed in the course of this work, and is very unlikely, but is not explicitly forbidden by the implementation of synaptic annealing proposed in this work.

power series. Thus in support of this thesis, a novel library was developed from efficiently generating numeric approximations of GSA distribution samples.

3.3.1.4 Uniform Synaptic Annealing Neighborhood.

3.3.2 Feed-Forward Neural Network Anisotropy Policies. In the course of developing the synaptic weight selection system presented in this thesis, it was observed that it is sometimes advantageous to apply different distributions to different synaptic weights.

3.4 Weight Space Traversal

In SA, the purpose of the neighborhood function is to control the traversal of the solution space of the problem. For synaptic annealing the solution space is the synaptic weight space, thus synaptic annealing neighborhood functions control the algorithms traversal of the weight space. The performance of a neighborhood function is entirely dictated by the weight space traversal, or walk, that it produces. In order to characterize the traversal properties of each neighborhood function, a traversal through a two-dimensional subset of the weight space will be analyzed for each neighborhood function. A small feed-forward neural network, with two input neurons, two hidden layer neurons, and a single output layer neuron, will be trained using synaptic annealing to solve a simple functional approximation problem. The network uses a $\tanh(\cdot)$ activation function, an initial annealing temperature of 10, and an α of 0.001. The function to be approximated is the complex-interaction function used in (22), which is given by

$$f_{CI}(u_1, u_2) = 1.9(1.35 + e^{0.5(u_1+1)} \sin(13(0.5u_1 - 0.1)^2) e^{0.5(u_2+1)} \sin(3.5u_2 + 3.5)). \quad (3.7)$$

The synaptic annealing algorithm will attempt to minimize the mean-squared error (MSE) cost function, as defined in section 3.5, on 100 randomly-drawn samples of f_{CI} . The validation error of the synaptic annealing algorithm will also be evaluated using an addition 100 randomly drawn samples, which are not presented to the algorithm during training. The same validation and training samples will be used for the analysis of each neighborhood

functions traversal. This section contains an analysis of the the weight space traversal properties of synaptic annealing during a relatively short training period. A more thorough analysis of the performance of synaptic annealing on this problem is presented in Chapter IV. For each neighborhood function, the traversal through the $W_{1,1,1}$ - $W_{2,2,1}$ space is analyzed. $W_{1,1,1}$ is the synaptic weight from input neuron 1 to hidden neuron 1, while $W_{2,2,1}$ is the synaptic weight from input neuron 2 to hidden layer neuron 2. The weight space traversal behavior of the algorithm was empirically found to be independent of the specific choice of weights. In the interest expedience and conciseness, in this section a traversal produced by a synaptic annealing algorithm using a neighborhood function for which the visiting distribution is x , will be called a x *traversal*.

3.4.1 Gaussian Neighborhood Function Weight Space Traversal. The weight space traversal exhibited by the Gaussian visiting distribution is consistent with a Gaussian random walk, as shown in Fig. 3.5(a). A single exemplary traversal is shown in Fig. 3.5; while the figure contains only one traversal, several identical experiments were conducted to ensure that the exemplar shown in the figure, as well as in figures 3.6, 3.7, 3.8, and 3.9, are typical of the execution of the algorithm.

Fig. 3.5(a) indicates that the traversal begins near the origin of the weight space and proceeds outward. The initial location is dictated by the values to which the weights in the feed-forward neural network are initialized, which are always random and small in magnitude in this work. The traversal proceeds to higher-magnitude weights as it follows the local cost surface of the training data set. Initially, most moves are accepted, indicated by a solid line in the figure, because the temperature is high during the first several training epochs. As the traversal proceeds and the temperature is lowered, the number of rejected moves, indicated by a dotted line, increases. The algorithm settles in a minima at $w_1 = 0.075$ $w_2 = 0.83$. Examining Fig. 3.5(b), a clear downward trend in training error, and upward trend in sum of squared weights, is present. It is also clear that there is no trend in the validation error, after an initial period of decline. Because large synaptic weights tend to cause their afferent neurons to behave like biases, these trends can be interpreted as the network memorizing the training data through the construction

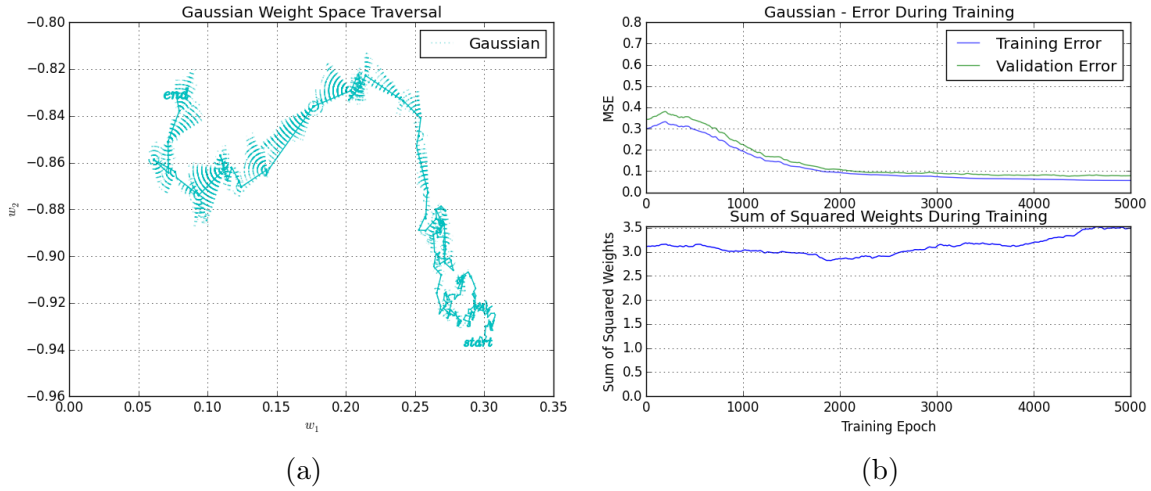


Figure 3.5 (a) A plot showing the traversal of the w_1, w_2 subspace of the weight space over the course of 5,000 epochs produced by a synaptic annealing algorithm employing a Gaussian visiting distribution. A solid line indicates a move which was accepted by the simulate annealing algorithm, while a dashed line indicates a move which was rejected. In this figure, only a few rejected moves are visible. The word *start* indicates the initial value of (w_1, w_2) , while the word *end* denotes the final value. (b) (upper) A plot showing the both the training and validation MSE of the results produced by the neural network in each epoch, smoothed using a central moving window average with a width of 21. This is the post-perturbation error, meaning that the error associated with moves that were rejected is shown. (lower) A plot showing the sum of squared weights of the neural network during each training epoch.

of fine-tuned biases. Thus, given sufficient time and sufficiently-many free parameters, the training error will become arbitrary low, while the validation error will remain relatively large.

3.4.2 Cauchy Neighborhood Function Weight Space Traversal. A neighborhood function based on a Cauchy visiting distribution produces a very different traversal pattern than one which uses a Gaussian visiting distribution. Inspection of Fig. 3.6(a) shows that a Cauchy visiting distribution causes a traversal which is prone to very long jumps through single dimensional of the weight space; note the difference in scale on the w_1 axis, relative to that of the traversal plot displayed in Fig. 3.5(a). This long-jump exploration behavior can be explained as a consequence of the shape of the Cauchy distribution. As can be seen in figures 3.2 and 3.3, the tails of the Cauchy distribution are longer than

those of the Gaussian distribution. This tail characteristic translates to a low, but non-zero probability of producing a long jump in a given dimension, thus enabling long jumps. However, because the probability of a long jump is low, it is unlikely to occur in both of the examined dimensions simultaneously, thereby producing the distinctive single-dimensional search pattern exhibited by the Cauchy visiting distribution. A closer inspection of the Fig. 3.6(a) reveals that the Cauchy visiting distribution produces a traversal which is similar to that of Gaussian distribution when the generated traversal distances remain near the mean.

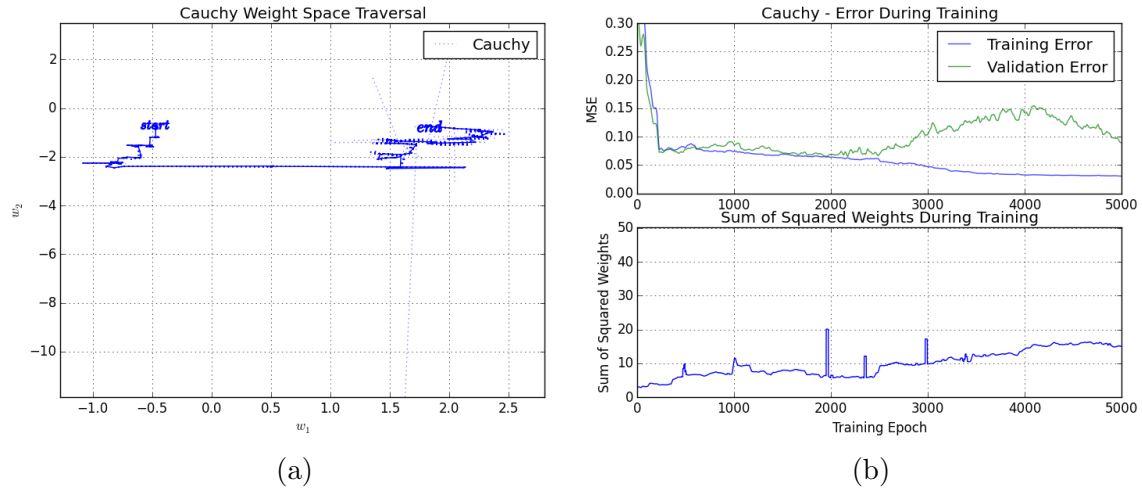


Figure 3.6 (a) A plot showing the traversal of the w_1, w_2 subspace of the weight space over the course of 5,000 epochs produced by a synaptic annealing algorithm employing a Cauchy visiting distribution. A solid line indicates a move which was accepted by the simulate annealing algorithm, while a dashed line indicates a move which was rejected. In this figure, only a few rejected moves are visible. The word *start* indicates the initial value of (w_1, w_2) , while the word *end* denotes the final value. (b) (upper) A plot showing the both the training and validation MSE of the results produced by the neural network in each epoch, smoothed using a central moving window average with a width of 21. This is the post-perturbation error, meaning that the error associated with moves that were rejected is shown. (lower) A plot showing the sum of squared weights of the neural network during each training epoch.

One can see how the traversal characteristics of a Cauchy visiting distribution effect the performance of a synaptic annealing algorithm by examining Fig. 3.6(b). Unlike the results generated by the Gaussian visiting distribution, the results in the upper plot of

Fig. 3.6(b) show a clear downward trend in both training and validation error. A complications associated with the use of a Cauchy visiting distribution for synaptic annealing is visible in Fig. 3.6(b) at approximately epoch 500. At this epoch a large increase in the sum of squared weights occurs, indicating that a significant change in the network occurred on this epoch. However, this change in the network configuration did not translate into a significant increase in the training error, likely because the afferent neuron to the modified synapse was already saturated, and thus was accepted by the algorithm. This change was, however, of great consequence to the validation error, which increased nearly threefold as a result.

3.4.3 Isotropic GSA Neighborhood Function Weight Space Traversal. The weight space traversal produced by a neighborhood function using an isotropically applied GSA visiting distribution is displayed in Fig. 3.7(a). This traversal reproduces a homogeneous pseudo-global search pattern indicative of GSA. As with the Cauchy visiting distribution neighborhood function, the traversal produced by the GSA distribution neighborhood function can be explained as a consequence of the distributions tail characteristics. The GSA distribution used to construct the traversal in Fig. 3.7(a) had the parameters $q_V = 2.5$, T_{q_V} , and $D = 1$. As can be seen in the lower plot of figures 3.2 and 3.3, the GSA distribution, using the parameters indicated, has significantly heavier and longer tails than the Cauchy distribution. This translates to a traversal which exhibits jumps that are longer and considerably more frequent, relative to a Cauchy traversal.

Given the observation, made in Sec. 3.4.2, that large jumps through the weight space have a tendency to create synaptic weights which cause their afferent neurons to behave as biases, it should be expected that the GSA traversal will readily produce such bias-like neurons. Fig. 3.7(b) provides evidence that this prediction is correct. The lower plot of Fig. 3.7(b) indicates that the sum of squared weights created by the GSA traversal surpasses the largest value observed in the Cauchy traversal at epoch 250. It should therefore be expected that a subsequent steep decrease in training error occurs; this is indeed what is observed in the upper plot of Fig. 3.7(b). Further, it should be expected that the validation error will decrease as well, in so far as the training set is representative of the entire data

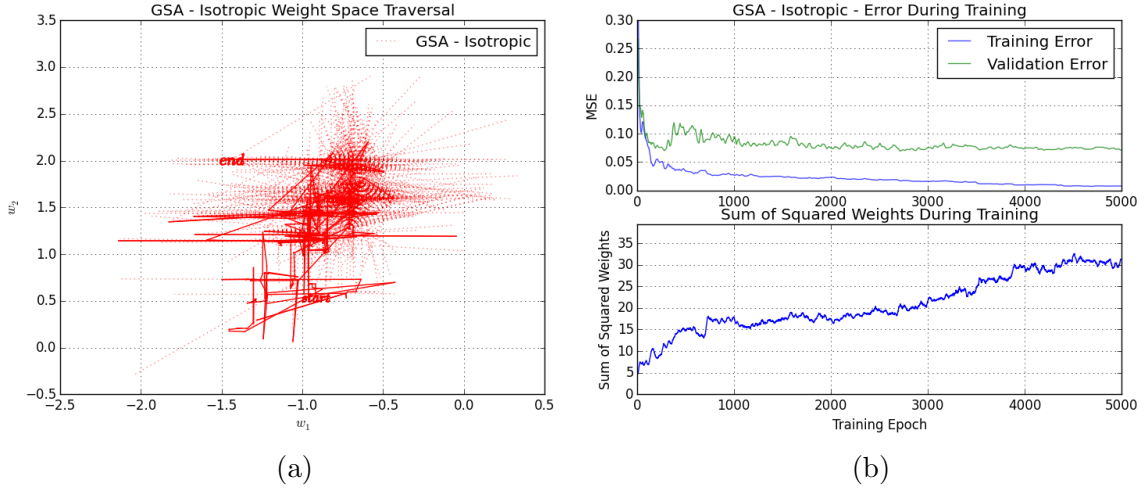


Figure 3.7 (a) A plot showing the traversal of the w_1, w_2 subspace of the weight space over the course of 5,000 epochs produced by a synaptic annealing algorithm employing an isotropic GSA visiting distribution. A solid line indicates a move which was accepted by the simulate annealing algorithm, while a dashed line indicates a move which was rejected. In this figure, only a few rejected moves are visible. The word *start* indicates the initial value of (w_1, w_2) , while the word *end* denotes the final value. (b) (upper) A plot showing the both the training and validation MSE of the results produced by the neural network in each epoch, smoothed using a central moving window average with a width of 21. This is the post-perturbation error, meaning that the error associated with moves that were rejected is shown. (lower) A plot showing the sum of squared weights of the neural network during each training epoch.

set, but not to the same extent as the training error; additionally, the validation error should gradually diverge from the training error as the sum of squared weight increases. These predicted characteristics, too, are present in Fig. 3.7(b). In order to reduce the generalization error introduced by these features of the GSA neighborhood function when applied to synaptic annealing, and anisotropic modification of the distribution is examined in the following section.

3.4.4 Weight-Anisotropic GSA Neighborhood Function Weight Space Traversal.

Fig. 3.8(a) is an example weight space traversal produced by a neighborhood function which uses a GSA distribution with weight anisotropy. As defined in Sec. 3.3.2, the weight anisotropy is an operator which is applied to the weight perturbation matrix. For each weight in the network, the operator shifts the mean of the visiting distribution for

that weight closer to the origin in proportion to the magnitude of the weight. Thus, the further a synaptic weight is from zero, the more the mean of its visiting distribution will be shifted toward zero. The effect of the weight anisotropy operator is to reduce the likelihood of weight space jumps which would result in a state which is far from the origin of the weight space. Contrasting Fig. 3.8(a) and Fig. 3.7(a), it is clear that the former depicts a traversal which has the homogeneous search characteristics of a GSA traversal, while simultaneously avoiding protracted traversals into large weight space values.

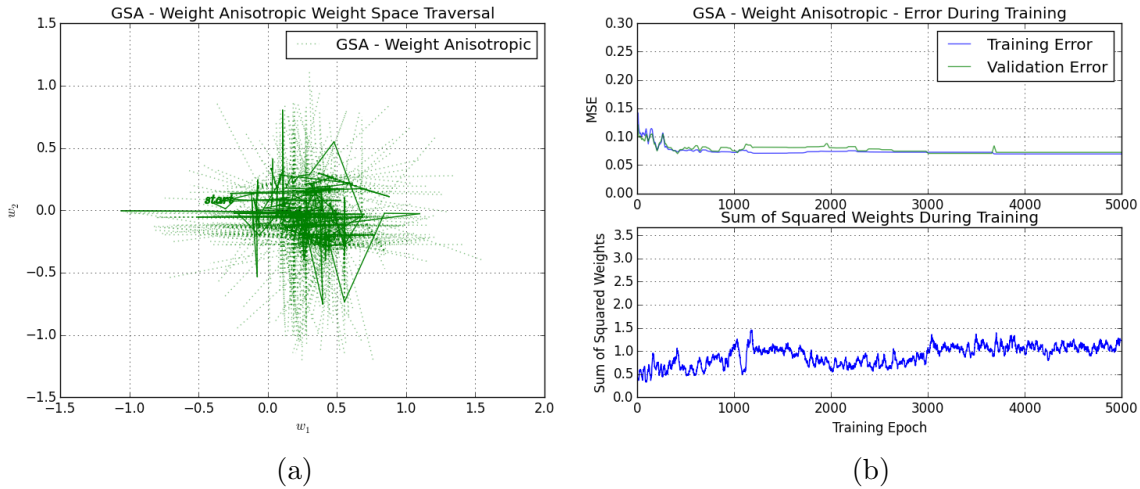


Figure 3.8 (a) A plot showing the traversal of the w_1, w_2 subspace of the weight space over the course of 5,000 epochs produced by a synaptic annealing algorithm employing a GSA visiting distribution with synaptic weight-based anisotropy. A solid line indicates a move which was accepted by the simulate annealing algorithm, while a dashed line indicates a move which was rejected. In this figure, only a few rejected moves are visible. The word *start* indicates the initial value of (w_1, w_2) , while the word *end* denotes the final value. (b) (upper) A plot showing the both the training and validation MSE of the results produced by the neural network in each epoch, smoothed using a central moving window average with a width of 21. This is the post-perturbation error, meaning that the error associated with moves that were rejected is shown. (lower) A plot showing the sum of squared weights of the neural network during each training epoch.

If properly constructed the weight anisotropy mechanism prevents weights from becoming large enough to saturate neurons in the network. Theoretically, this is a desirable characteristic for a neighborhood function as it will prevent any very large weights from arising in the network, which prevents the creation of saturated neurons. A network

with less saturated neurons acting as biases will be less biased, by definition. A synaptic annealing algorithm using a weight anisotropic GSA neighborhood function should therefore be expected to yield lower validation set errors, relative to an algorithm which uses an isotropic GSA neighborhood function. In the upper plot of Fig. 3.8(b) the validation and training error for the traversal shown in Fig. 3.8(a) are displayed. In contrast to the preceding weight space traversal performance plots, the training and validation error of the weight anisotropic GSA traversal are found to be in close agreement. Additionally, the lower plot of Fig. 3.8(b) indicates that there is no significant trend the the sum of squared weight values for the network, which is also a property unique to the weight anisotropic GSA neighborhood function. These results indicate that weight anisotropy may serve as an alternative weight minimization technique to the multiobjective simulated annealing approach described in (22).

3.4.5 Uniform Neighborhood Function Weight Space Traversal. The weight space traversal produced by a uniform neighborhood distribution, show in Fig. 3.9(a), is very similar to that of a Gaussian neighborhood function. The most significant difference between the traversals is total area of the weight space explored. The Gaussian traversal covers considerably more area than the Uniform traversal. This behavioral difference is the result of the fact that the uniform distribution is over the range $[-\frac{1}{2}, \frac{1}{2}]$, which is much smaller than the domain of the standard normal distribution used in the Gaussian distribution.

Examining the training error performance shown in Fig. 3.9(b), it is clear that the exemplar uniform neighborhood function significantly under-performed, with respect to the Gaussian neighborhood function. This deficiency can be explained primarily as a consequence of the relatively limited range of the uniform distribution. The restricted search range combined with the learning rate used in these traversal experiments yield very little change in the state of the network, as illustrated in the sum of squared weights plot displayed in the lower pane of Fig. 3.9(b).

3.4.6 Comparative Weight Space Traversal. It is instructive to directly compare the traversals produced by the various distributions considered in the preceding sections.

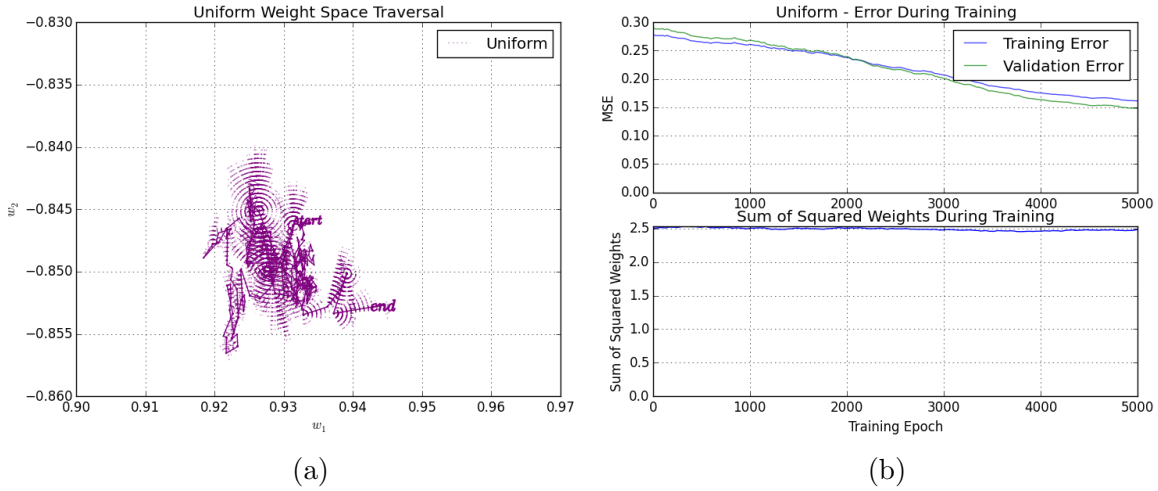


Figure 3.9 (a) A plot showing the traversal of the (w_1, w_2) subspace of the weight space over the course of 5,000 epochs produced by a synaptic annealing algorithm employing a visiting distribution which is uniform over the range $[-\frac{1}{2}, \frac{1}{2}]$. A solid line indicates a move which was accepted by the simulate annealing algorithm, while a dashed line indicates a move which was rejected. In this figure, only a few rejected moves are visible. The word *start* indicates the initial value of (w_1, w_2) , while the word *end* denotes the final value. (b) (upper) A plot showing the both the training and validation MSE of the results produced by the neural network in each epoch, smoothed using a central moving window average with a width of 21. This is the post-perturbation error, meaning that the error associated with moves that were rejected is shown. (lower) A plot showing the sum of squared weights of the neural network during each training epoch.

Fig. 3.10 displays these traversals on a common weight space; several interesting traversal characteristics are present in the figure. The impact of the weight anisotropy on the traversal characteristics of the synaptic annealing algorithm are demonstrated by the difference in the weight-space location in which the respective searches occur: The weight anisotropic traversal is limited to a range near the origin, while the isotropic traversal searches a higher magnitude region of the weight space. The relatively diminutive total traversal length of both the Gaussian and uniform neighborhood functions are also apparent when compared to the range of the traversals produced by the other visiting distributions.

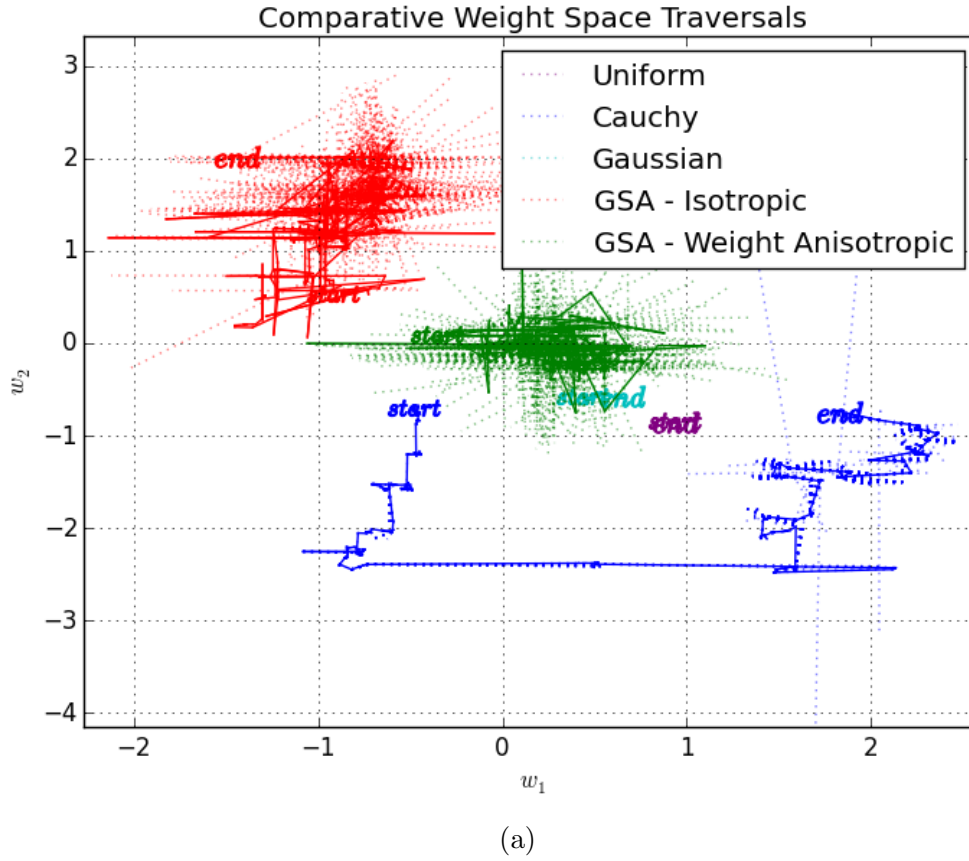


Figure 3.10 (a) A plot juxtaposing the weight space traversals of produced by a synaptic annealing algorithm employing several different visiting distributions.

The preceding examination of weight space traversals is intended to serve as a exposition of the characteristics of the synaptic annealing algorithm when coupled with various visiting distributions, and is not explicitly concerned with a rigorous statistical examination of the performance of the algorithm. It is, however, instructive to explicitly compare the performance of each of the exemplar traversals discussed in this section to ensure consistency with behavior expectations. The regression error performance of the each neighborhood function is displayed in Table 3.1.

Examining Table 3.1 suggests a few trends. As would be expected, the training set MSE of GSA is considerably lower than that of FSA (Cauchy visiting distribution), which is in turn better than CSA (Gaussian visiting distribution). These findings are consistent

Table 3.1 Training and Validation Set Mean Squared Error at Epoch 5,000

Neighborhood Function Visiting Distribution	Training Error	Validation Error
Gaussian	0.05549	0.07743
Cauchy	0.03028	0.08878
Isotropic GSA	0.00739	0.07202
Weight Anisotropic GSA	0.06947	0.07229
Uniform	0.16111	0.14773

with previous findings (33, 10) regarding the efficiency of various SA implementations. Interestingly, the validation set MSE, which in this case is specific to the problem of feed-forward neural network synaptic weight selection, does not follow this trend. While it may seem that introducing weight anisotropy only served to increase the final training set MSE of the synaptic annealing algorithm, the mechanism that produced this increase in error is actually advantageous, as discussed in Sec. 3.4.4.

3.5 Cost Functions

All variations of synaptic annealing require the specification of a heuristic cost function, which conforms to the specification given in section 3.2. In this thesis, the synaptic annealing algorithm will be applied to two broad problem classes: function approximation and classification. To apply synaptic annealing to both of these problem classes, two cost functions must be constructed.

3.5.1 Regression Error Cost Function. The cost function chosen to serve as a heuristic of minimum regression error in this thesis is given by

$$\mathcal{C}_r(\mathbf{W}, X) = \frac{\sqrt{\sum [\lambda - \Phi(\mathbf{W}, \chi)]^2}}{|X|} \quad (3.8)$$

where \mathbf{W} is a weight matrix, Φ is the propagation function, and X is the set of ordered pairs mapping a set of input data χ to a set of output values λ , in which χ is the first entry and λ is the second entry. This cost function yields the mean Euclidean between the desired output set λ and the output of the forward propagation, $\Phi(\mathbf{W}, \chi)$.

3.5.2 Classification Error Cost Function. In this thesis, the classification cost function, \mathcal{C}_c , yields the number of incorrectly classified observations in a given data set, which is given mathematically by

$$\mathcal{C}_c(\mathbf{W}, X) = \sum (\lambda \neq \Phi(\mathbf{W}, \chi)). \quad (3.9)$$

where \mathbf{W} is a weight matrix, Φ is the propagation function, and X is the set of ordered pairs mapping a set of input data χ to a set of output values λ , in which χ is the first entry and λ is the second entry. This function will be used to evaluate the performance of a network configuration, for a given set of data, when that data consists of categorical, or labeled, data. Though the classification cost function will always be used to report the performance of a network configuration when applied to classification data, it may or may not be used as the cost function which informs the synaptic annealing algorithm. Often, it is advantageous to train a neural network using the regression error function (13) presented in section 3.5.2, and then report the performance of that network using the classification error.

3.6 Data

The performance of the synaptic annealing methodology described in this chapter will be evaluated in the following chapter. In order to ensure that the methodology is generally applicable, it will be applied to several different data sets which vary in complexity and organization.

3.6.1 Classification Data Sets. Three classification data sets will be used in this thesis:

3.6.2 Function Approximation Data Sets. Function approximation performance will be evaluated using the complicated interaction function, which is given mathematically by the formula

$$f_{CI}(u_1, u_2) = 1.9(1.35 + e^{0.5(u_1+1)} \sin(13(0.5u_1 - 0.1)^2) e^{0.5(u_2+1)} \sin(3.5u_2 + 3.5)), \quad (3.10)$$

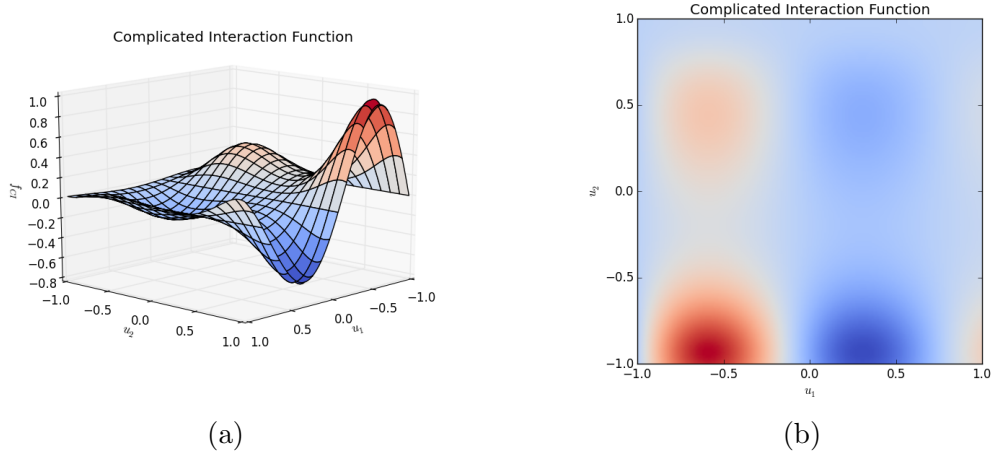


Figure 3.11 (a) A plot displaying the surface produced by the complicated interaction function, f_{CI} , in two dimensions, u_1 and u_2 (b) A color contour plot of the complicated interaction function.

and is shown in Fig. 3.11

3.7 Design of Experiments

IV. Results

4.0.1 Classification Performance.

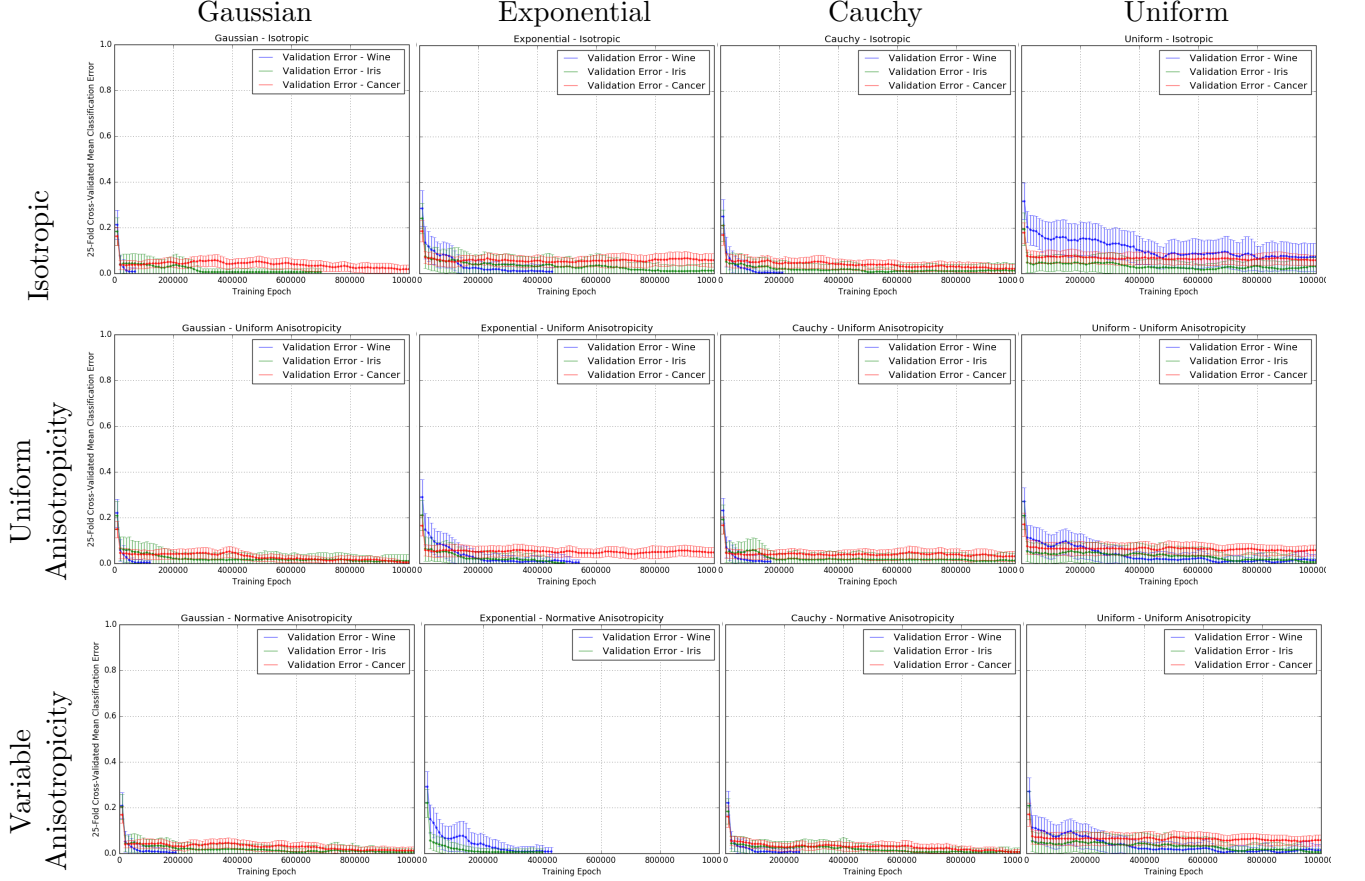


Figure 4.1 The simulation-time evolution of the 25-fold cross-validated classification error of a feed-forward neural network with 50 hidden units on Fishers iris data. Each column in this table depicts the results for an SA neighborhood function employing a single type of visiting distribution, which is indicated in the column header. Each row depicts a single type of anisotropy strategy, which is indicated in the row header. The classification error is defined as the fraction of samples incorrectly classified.

4.0.2 Function Approximation Performance.

V. Conclusion

We have proposed anisotropy as a new simulated annealing neighborhood function parameter, presented a framework for applying simulated annealing to feed forward neural network weight selection, and numerically analyzed the performance of annealing systems constructed according to this framework. We have demonstrated that the anisotropy of a neighborhood function can have a significant impact on the classification performance of a feed forward neural network.

An exploratory study of oscillatory variation in the distribution temperature parameter of the SGSA distribution yielded encouraging preliminary results. Further study may yield additional insights into the benefits of alternating global and local distribution characteristics during a semi-local solution space search.

Introducing a greedy local optimization algorithm

Appendix A. First appendix title

A.1 In an appendix

This is appendix section A.1.

Note: I highly recommend you create each chapter in a separate file including the `\chapter` command and `\include` the file. Then you can use `\includeonly` to process selected chapters and you avoid having to latex/preview/print your entire document every time.

Bibliography

1. Anderson, James A. "A Simple Neural Network Generating an Interactive Memory," *Mathematical Biosciences*, 14:197–220 (1972).
2. Anderson, James A. and E. Rosenfeld, editors. *Neurocomputing: Foundations of Research*. Cambridge, MA: MIT Press, 1988.
3. Andricioaei, Ioan and John E. Straub. "Generalized simulated annealing algorithms using Tsallis statistics: Application to conformational optimization of a tetrapeptide," *Phys. Rev. E*, 53:R3055–R3058 (Apr 1996).
4. Barto, Andrew G., et al. "Artificial Neural Networks." edited by Joachim Diederich, chapter Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems, 81–93, Piscataway, NJ, USA: IEEE Press, 1990.
5. Block, H. D. "The Perceptron: A Model for Brain Functioning," *Reviews of Modern Physics*, 34:123–135 (1962).
6. Brodie, S. E., et al. "The response of the Limulus retina to moving stimuli: a prediction by Fourier synthesis," *J Gen Physiol*, 72(2):129–66 (August 1978).
7. Bryson, A. E. and Y. C. Ho. *Applied Optimal Control*. New York: Blaisdell, 1969.
8. Corana, A., et al. "Minimizing Multimodal Functions of Continuous Variables with the "Simulated Annealing" algorithmCorrigenda for This Article is Available Here," *ACM Trans. Math. Softw.*, 13(3):262–280 (September 1987).
9. Cybenko, G. "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, 2(4):303–314 (1989).
10. Dall'Igna J, Alcino, et al. "Performance and parameterization of the algorithm Simplified Generalized Simulated Annealing," *Genetics and Molecular Biology*, 27:616 – 622 (00 2004).
11. Engel, Jonathan. "Teaching Feed-forward Neural Networks by Simulated Annealing," *Complex Syst.*, 2(6):641–648 (December 1988).
12. Goffe, William L., et al. "Global Optimization of Statistical Functions with Simulated Annealing," *Journal of Econometrics*, 60:65–99 (1994).
13. Haykin, Simon. *Neural networks: a comprehensive foundation*. Upper Saddle River, N.J: Prentice Hall, 1999.
14. Hebb, D. O. *The organization of behavior; a neuropsychological theory*, (by) D.O. Hebb. Science Editions. New York: John Wiley and Sons, 1967.
15. Hopfield, J. J. "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, 79 (1982).
16. Ingber, Lester, "Very Fast Simulated Re-Annealing," 1989.

17. James, W. *The Principles of Psychology*. Number v. 1 in American science series: Advanced course, H. Holt, 1890.
18. Kirkpatrick, S., et al. "Optimization by simulated annealing," *SCIENCE*, 220(4598):671–680 (1983).
19. Kohonen, Teuvo. "Correlation Matrix Memories," *IEEE Trans. Comput.*, 21(4):353–359 (April 1972).
20. Lecchini-Visintini, A., et al. "Simulated Annealing: Rigorous finite-time guarantees for optimization on continuous domains," *ArXiv e-prints* (September 2007).
21. Lecun, Y. "Une procédure d'apprentissage pour réseau à seuil asymétrique," *Proceedings of Cognitiva 85, Paris*, 599–604 (1985).
22. Lee, Yeejin, et al. "Improving generalization capability of neural networks based on simulated annealing.." *IEEE Congress on Evolutionary Computation*. 3447–3453. IEEE, 2007.
23. McCulloch, Warren S. and Walter Pitts. "Neurocomputing: Foundations of Research." edited by James A. Anderson and Edward Rosenfeld, chapter A Logical Calculus of the Ideas Immanent in Nervous Activity, 15–27, Cambridge, MA, USA: MIT Press, 1988.
24. Metropolis, N., et al. "Equation of State Calculations by Fast Computing Machines," 21:1087–1092 (June 1953).
25. Minsky, M. and S. Papert. *Perceptrons*. Cambridge, MA: MIT Press, 1969.
26. Mukherjee, S. and B. K. Chakrabarti. "Multivariable optimization: Quantum annealing and computation," *European Physical Journal Special Topics*, 224:17 (February 2015).
27. Parker, D. B. *Learning Logic*. Technical Report TR-47, Cambridge, MA: Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, 1985.
28. Piccinini, Gualtiero. "Computational Explanation in Neuroscience," *Synthese*, 153(3):343–353 (2006).
29. Rochester, N., et al. "Tests on a cell assembly theory of the action of the brain, using a large digital computer," *Information Theory, IRE Transactions on*, 2(3):80–93 (September 1956).
30. Rosenblatt, F. "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, 65(6):386–408 (1958).
31. Rumelhart, D.E., et al. *Learning Internal Representations by Error Propagation*. 1986.
32. Szu, Harold and Ralph Hartley. "Fast simulated annealing," *Physics Letters A*, 122(3-4):157–162 (June 1987).
33. Tsallis, Constantino and Daniel A. Stariolo. "Generalized simulated annealing," *Physica A: Statistical and Theoretical Physics*, 233(1-2):395–406 (November 1996).

34. von der Malsburg, Chr. "Self-Organization of Orientation Sensitive Cells in the Striate Cortex,"
35. Werbos, P. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD dissertation, Harvard University, 1974.
36. Widrow, Bernard and Marcian E. Hoff. "Adaptive Switching Circuits." *1960 IRE WESCON Convention Record, Part 4*. 96–104. New York: Institute of Radio Engineers, 8 1960.

Vita

Insert your brief biographical sketch here. Your permanent address is generated automatically.

Permanent address: 452 Orchard Drive
Oakwood, Ohio 45419