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 vi

List of Tables vii

List of Symbols viii

List of Abbreviations ix

Abstract x

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 2-10

2.1.4 Activation Functions 2-10

2.1.5 Learning Strategies 2-11

2.2 Related Works in Simulated Annealing 2-11

2.2.1 Reheating 2-13

2.3 Related Works in Quantum Mechanics 2-13

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
3.2 Neighborhood Functions: Traversing the Cost Surface 3-1
3.2.1 Visiting Distributions 3-1
3.2.2 Anisotropicity Policies 3-4
3.3 Feed-Forward Neural Network Representation 3-5
3.4 Applying Anisotropic SA to Feed-Forward Neural Networks . 3-6

IV. Results 4-1

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. (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. 2-14

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. 3-3

3.2. (a) An arbitrary feed-forward ANN. (b) The weight matrix representation of the feed-forward ANN in (a). 3-4

List of Tables

Table Page

List of Symbols

Symbol Page

List of Abbreviations

Abbreviation Page

ANN Artificial Neural Network ix

1

Simulated Annealing2-13

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 (11) 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 in 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 the 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 neurons activation function is just a phenomenon emerges from 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 (19) 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 atomata (24). In (19) 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 (19) 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 posses 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. (25) 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 Hebb[1]. 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 (12) 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 (26). 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

[1]It 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 (14) in Chapter XVI of *Psychology (Briefer Course)*.

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:

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

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 from 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 (4). 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:

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

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 it 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 a error gradient descent learning rule (32), as described by Widrow and Hoff.

In 1969 Minksy 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 the 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 (21). 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 (16) 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 (30) 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 linear[2] model and the biological output signals was found to be in excellent agreement (5). This finding was interesting for the purposes of perceptron simulation, but was ultimately found not not to hold for larger collections of neurons.

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

[2]Linear, 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.

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 (13). 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

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

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

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

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 reached[3]. 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 patterns[4]. 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 Hopfield[5]. 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 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

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

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

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

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

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

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

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(3). 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 (11).

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 (6) (31), and the method was simultaneously discovered independently by two other groups (23) (18), it was Rumelhart et. al. that applied the algorithm to machine learning (27). Back-propagation can be thought of as a generalization of the gradient descent[6] algorithm presented by Widrow and Hoff (32), 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 (21).

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 in principle capable of approximating any continuous function (8). 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.

[6]The 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.

2.1.5 Learning Strategies. Stub.

2.1.5.1 Back Propagation Training. Stub.

2.1.5.2 Simulated Annealing. Stub.

2.2 Related Works in Simulated Annealing

Simulated annealing (SA) 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 (15) 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 (20) 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 (10), Corana et al. (7), and Lecchini-Visintini et. al. (17) extended SA to the continuous domain.

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

$$\mathcal{C} : \boldsymbol{\mathcal{S}} \to \mathbb{R}.$$

Because $\mathcal{C}$ is a function on $\boldsymbol{\mathcal{S}}$, it is said that the cost function forms a cost surface in the solution space. During each iteration of the algorithm, a new solution, $\boldsymbol{s}'$, is generated using

$\boldsymbol{\mathcal{N}}(\boldsymbol{s})$, and the cost of that solution, $\mathcal{C}(\boldsymbol{s}')$, is determined. The change in cost associated with moving from the current solution to the neighbor solution is given by

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

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 (15) 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 (20), given by:

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

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:

$$\mathcal{P}_t = e^{-\frac{\Delta V}{T}} \tag{2.8}$$

The SA algorithm is presented in Algorithm 1.

Algorithm 1 Simulated Annealing

$\boldsymbol{s} \leftarrow \boldsymbol{s}_0$
$t \leftarrow 0$
while $T(t) > \epsilon$ **do**
  $\boldsymbol{s}' \leftarrow \boldsymbol{\mathcal{N}}(\boldsymbol{s})$
  $\Delta C \leftarrow (\mathcal{C}(\boldsymbol{s}') - \mathcal{C}(\boldsymbol{s}))$
  if $\Delta C \leq 0$ **then**
    $\boldsymbol{s} \leftarrow \boldsymbol{s}'$
  else $\{\exp(\Delta C/T(t)) > U(0,1)\}$
    $\boldsymbol{s} \leftarrow \boldsymbol{s}'$
  end if
  $t \leftarrow t + 1$
end while
$\boldsymbol{s}_{opt} \leftarrow \boldsymbol{s}$
return $(\boldsymbol{s}_{opt})$

We need here a discussion regarding the asymptotic convergence.

In (28) 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 (29), generalized both CSA and FSA into a single framework: generalized simulated annealing (GSA), which was faster still.

2.2.1 Reheating.

Stub.

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:

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

where Γ is the tunneling field strength (22). Fig. 2.1 depicts a one-dimensional example of quantum tunneling.

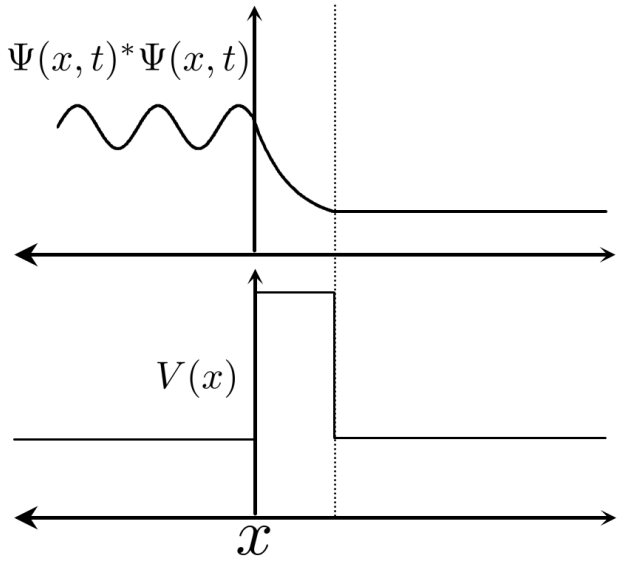


Figure 2.1 (Bottom) A simple step potential in one dimension. (Top) The probability density function of a generic quantum system in the presence of a potential energy step barrier. There is an exponential decrease in probability through the barrier, and a uniform probability beyond the barrier.

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 development of several formalisms

3.1 Simulated Annealing

3.2 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 search 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 (28, 29), and is often specified using a *visiting distribution*, which is defined as a probability distribution over the move distance parameter. The visiting distribution specifies the magnitude of the move, but does not specify anything about the direction of the move.

The direction of a cost surface traversal is the allocation of the total move distance to each of the possible traversal dimensions. A traversal dimension, or degree of freedom, is a solution parameter which can be modified by the SA algorithm. In previous work [need several citations here](), 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 total distance moved in each of the possible travel dimensions is equal in magnitude, and random in sign. In the following sections a novel method is developed for applying the cost surface traversal distance anisotropically.

3.2.1 Visiting Distributions.

The visiting distribution of a neighborhood function is a probability distribution function over all possible traversal distances. Once a visiting

distribution is specified, samples can be drawn from the distribution using inverse transform sampling. During each iteration of the SA algorithm, a sample is drawn from the visiting distribution to determine the solution-space distance to be traversed during that iterations. In the following sections several visiting distributions, and the corresponding sample generation functions, will be discussed.

3.2.1.1 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) (29).

3.2.1.2 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 document to describe any distribution possessing this property.

In (28) 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 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. Therefore, 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.

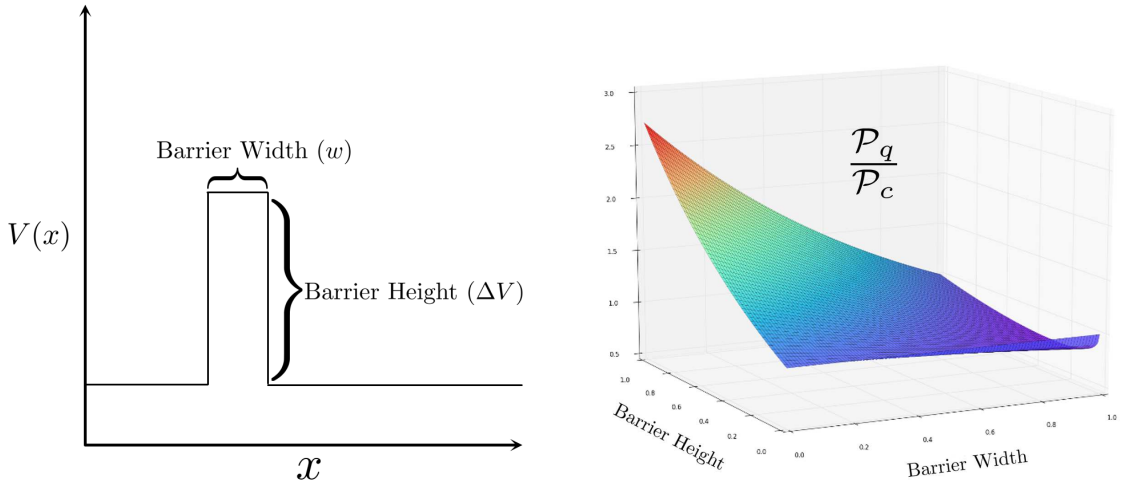
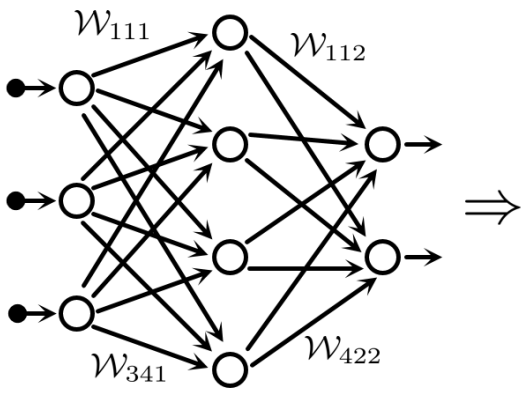


Figure 3.1 (a) A potential energy barrier on a one-dimensional potential energy surface. (b) The tunneling probability relative to the probability of traversal due to thermal fluctuation for a step barrier plotted as a function of the height and width of the barrier.

3.2.1.3 Exponential Visiting Distributions.



(a)

$$\boldsymbol{\mathcal{W}}_{k=1} = \begin{bmatrix} \mathcal{W}_{111} & \mathcal{W}_{121} & \mathcal{W}_{131} & \mathcal{W}_{141} \\ \mathcal{W}_{211} & \mathcal{W}_{221} & \mathcal{W}_{231} & \mathcal{W}_{241} \\ \mathcal{W}_{311} & \mathcal{W}_{321} & \mathcal{W}_{331} & \mathcal{W}_{341} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{\mathcal{W}}_{k=2} = \begin{bmatrix} \mathcal{W}_{112} & \mathcal{W}_{122} & 0 & 0 \\ \mathcal{W}_{212} & \mathcal{W}_{222} & 0 & 0 \\ \mathcal{W}_{312} & \mathcal{W}_{322} & 0 & 0 \\ \mathcal{W}_{412} & \mathcal{W}_{422} & 0 & 0 \end{bmatrix}$$

(b)

Figure 3.2 (a) An arbitrary feed-forward ANN. (b) The weight matrix representation of the feed-forward ANN in (a).

3.2.1.4 Uniform Visiting Distributions. As can be seen in Fig. 2.1 the probability of observing a particle past a classically-impenetrable potential barrier is uniform beyond the barrier.

3.2.2 Anisotropicity Policies. Anisotropic application policies are equivalent to drawing from a multidimensional sampling distribution, with each dimension having a tunable shape parameter. The anisotropicity term in effect transforms the distribution.

3.2.2.1 Isotropic Anisotropicity. Applying an isotropic anisotropicity policy is equivalent to drawing the sample movement from an n-dimensional visiting distribution over an n-dimensional solution space.

3.2.2.2 Uniform Anisotropicity. The uniform anisotropicity policy draws the anisotropicity specification from a uniform distribution. The amount of the total traversal distance applied to each dimension is drawn from a uniform distribution over a constant range such as $[0, 1]$

3.2.2.3 Variable Anisotropicity.

3.3 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 variation 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 connection. 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. 3.2. The weight matrix, $\boldsymbol{\mathcal{W}}$, therefore encodes a configuration in the solution space of the problem, and can be visualized as:

$$\boldsymbol{\mathcal{W}} = \begin{bmatrix} \mathcal{W}_{111} & \mathcal{W}_{121} & \dots & \mathcal{W}_{1j1} \\ \mathcal{W}_{211} & \mathcal{W}_{221} & \dots & \mathcal{W}_{2j1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{W}_{i11} & \mathcal{W}_{i21} & \dots & \mathcal{W}_{ij1} \end{bmatrix} \begin{bmatrix} \mathcal{W}_{112} & \mathcal{W}_{122} & \dots & \mathcal{W}_{1j2} \\ \mathcal{W}_{212} & \mathcal{W}_{222} & \dots & \mathcal{W}_{2j2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{W}_{i12} & \mathcal{W}_{i22} & \dots & \mathcal{W}_{ij2} \end{bmatrix} \cdots \begin{bmatrix} \mathcal{W}_{11k} & \mathcal{W}_{12k} & \dots & \mathcal{W}_{1jk} \\ \mathcal{W}_{21k} & \mathcal{W}_{22k} & \dots & \mathcal{W}_{2jk} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{W}_{i1k} & \mathcal{W}_{i2k} & \dots & \mathcal{W}_{ijk} \end{bmatrix}$$

The total solution space, $\boldsymbol{\mathcal{S}}$, may then be defined as the set of all possible configurations of $\boldsymbol{\mathcal{W}}$ for a given neural network. In the synaptic weight selection problem domain it is more evocative to call $\boldsymbol{\mathcal{S}}$ the weight space of the network, so this convention adopted throughout this thesis. Given $\boldsymbol{\mathcal{S}}$, we define a cost function $\mathcal{C}$ to be the mapping

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

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

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

With this framework in place, the SA can be applied to transition from a randomly-selected initial state, $\boldsymbol{\mathcal{W}}_0$, to $\boldsymbol{\mathcal{W}}_{opt}$.

3.4 Applying Anisotropic SA to Feed-Forward Neural Networks

With the feed-forward neural network problem representation described in sections 2.6-2.6, and the description of SA

We can map the concept of a solution space, discussed in Sec. **??**, to the weight space of the feed-forward neural network. It is also clear that the cost function is an error function defined using the correct output value and the predicted output value.

3.4.0.4 Classical Neighborhood Functions. We define $\mathcal{N}_c$ to be the classical neighborhood function for feed-forward neural network weights, which is defined as

$$\mathcal{N}_c(\boldsymbol{\mathcal{W}}) = \boldsymbol{\mathcal{W}} + \alpha g_G(u)\boldsymbol{\mathcal{A}} \tag{3.1}$$

where α is the learning rate, $g_G(u)$ is a Gaussian generating function, u is a uniform random variable over the range $U[0,1]$, and $\boldsymbol{\mathcal{A}}$ is a matrix with dimensionality equal to that of $\boldsymbol{\mathcal{W}}$. Each element of $\boldsymbol{\mathcal{A}}$ is generated from a distribution over the range $[-1,1]$. $\boldsymbol{\mathcal{A}}$ is restricted such that for each element of $\boldsymbol{\mathcal{W}}$ that is zero the corresponding element in $\boldsymbol{\mathcal{A}}$ is zero, and that $\boldsymbol{\mathcal{A}}$ must be normalized such that the magnitude of the matrix elements sum to 1. These restrictions ensure two useful properties of the classical neighborhood function, that:

1. $\mathcal{N}_c(\boldsymbol{\mathcal{W}})$ will produce a weight matrix, $\boldsymbol{\mathcal{W}'}$, which will have an L^2 distance of exactly $\alpha g_G(u)$ from the original matrix, $\boldsymbol{\mathcal{W}}$, in the weight space. Said differently, no matter how the total change in the weight matrix is distributed among the weights, the total distance of the change is conserved.
2. This traversal distance will be distributed anisotropically over the weights, with the anisotropicity determined by the distribution used to generate $\boldsymbol{\mathcal{A}}$. (See Sec. **??** for more information.)

These properties are useful in that they allow for strict control over the systems traversal of the cost surface, and the decouple the implementations of the visiting distribution and anisotropicity.

The neighborhood function given in Eq. **??** 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. **??**, 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 next section, a model which approximates quantum mechanical phenomena will be constructed.

3.4.0.5 Quantum-Inspired Neighborhood Functions. The classical neighborhood function present in Sec. **??** is one of many which could be employed in CSA. In order to incorporate quantum tunneling into this model, we must have some mechanism which allows the neighborhood function to generate neighbor states which are across a cost surface barrier. Since it is impossible to know the cost function value of a configuration which has not yet been evaluated, we must construct a neighborhood function which is able to jump to these configurations. This mechanism is often called a trial jump. We define our quantum neighborhood function for feed-forward neural network weights to be

$$\mathcal{N}_q(\boldsymbol{\mathcal{W}}) = \boldsymbol{\mathcal{W}} + \alpha g_q(u)\boldsymbol{\mathcal{A}} \tag{3.2}$$

where α, u, and $\boldsymbol{\mathcal{A}}$ are defined as they are in Eq. **??**, and $g_q(u)$ is a parameterized generation function used to produce a value from the visiting distribution. As in (29), the visiting distribution is defined as the probability distribution function of the trial jump distance.

Several visiting distributions may be used used to approximate quantum tunneling in order to form a quantum neighborhood function, g_q. The most obvious choice is the

exponential distribution, for which the generation function is given by

$$g_e(u) = \frac{-\ln(u)}{\gamma} \tag{3.3}$$

where γ, the scale parameter for the distribution, is defined as Γ. The exponential distribution closely aligns with physical reality, as the probability of penetrating a barrier decreases exponentially with the barriers width. This can be seen in Eq. **??** and Fig. 2.1. The barrier width in Eq. **??** and Fig. 2.1 is analogous to the trial jump distance in the quantum neighborhood function. The introduced stochastic control parameter Γ denotes the strength of the tunneling field as in (22), and thus controls the likelihood of a long trial jump. Γ is supported on $[0, 1]$. A large value of Γ corresponds to frequent, long range quantum trial jumps. It is the stochastic control parameter Γ value that connects the quantum neighborhood function to quantum annealing. Much like FSA (28), this provides mostly local search with occasional global-scale searches. Other generating functions corresponding to other visiting distributions may also be used to generate new states.

In FSA the Cauchy distribution is used as the visiting distribution. The generation function for the Cauchy distribution is

$$g_C(u) = c \tan\left(u - \frac{1}{2}\right) \tag{3.4}$$

where c is the shape parameter of the Cauchy distribution used for the visiting distribution. We set c to be $1/(1-\Gamma)$, which ensures that $g_C(u) \to u\,|\boldsymbol{\mathcal{S}}|$, where $|\boldsymbol{\mathcal{S}}|$ is the total size of the configuration space, in the $\Gamma \to 1$ limit. That is, as Γ, the stochastic control parameter for tunneling, goes to it's maximum value, $g_C(u)$ approaches the uniform distribution, thereby allowing tunneling of any distance.

Finally, we consider the case of the uniform visiting distribution, which has an equal probability of visiting any possible configuration in the solution space. Inspecting Fig. 2.1 we see that the probability of tunneling to any location past the step potential energy

barrier is uniform[1] . If we assume that every cost surface barrier encountered by the algorithms is small relative to the size of the complete cost surface, it is not unreasonable to approximate the probability of transition to any state as uniform. The generation function for this distribution is given by

$$g_U(u) = (g_G(u) + u\,|\boldsymbol{\mathcal{S}}|\,(\Gamma \geq u)) \tag{3.5}$$

which results in a trial jump to any location in the configuration space, but only with probability equal to Γ. In the case where a uniform trial jump does not occur, $g_U(u)$ reduces to $g_G(u)$. In other words, in the $\Gamma \to 0$ limit, CSA is recovered.

3.4.0.6 Anisotropicity. In the course of developing the synaptic weight selection system presented in this paper, it was observed that it is sometimes advantageous to move along an error surface in a relatively small subset of the total number of dimensions. Such a modification of the configuration corresponds to the modification of a single synaptic weight in a training epoch. Though it is possible that this could occur by chance, the likelihood of a single-weight modification is inversely proportional to the number of weights in the network. For large networks this probability becomes vanishingly small, so a mechanism is introduced to increase the frequency of occurrence of these fine-tuning events. In the physical science a directionally-dependent event, one that varies differently in some subset of its degrees of freedom, is called an anisotropy. Thus, an anisotropicity matrix, $\boldsymbol{\mathcal{A}}$, which indicates how much of a given configuration change will be applied to each of the weights, which are analogous to degrees of freedom in the neural network configuration. The restrictions described in Sec. **??** hold. In this section we turn our attention to the mechanism used to generate the distribution of values in $\boldsymbol{\mathcal{A}}$.

The simplest possible realization of $\boldsymbol{\mathcal{A}}$ is the isotropic limit. In this case the total L^2 distance traversed through the weight space is distributed evenly among the weights. We

[1]The quantum mechanical analogy is intentionally incomplete here. Strictly speaking, the probability is only uniform over the entire cost surface if no other barriers exist on the surface

denote the anisotropicity matrix corresponding to this case as

$$\boldsymbol{\mathcal{A}}_0 = \frac{1}{D}\boldsymbol{J}_{\pm} \tag{3.6}$$

where D is the number of non-zero elements in $\boldsymbol{\mathcal{W}}$, and $\boldsymbol{J}_{\pm}$ is a unit matrix of equal dimensionality to $\boldsymbol{\mathcal{W}}$ in which each element is randomly and with equal probability assigned a sign. This definition of anisotropicity scales the step size inversely with the number of synaptic connection in the matrix. In very large matrices, this is problematic because the change in each individual synaptic weight becomes so small that the change in weight value produces effectively no change in the cost function value, thus causing the cost surface to flatten. In practice, we observe that it is possible to counteract this gradient dilution by simply scaling the anisotropicity matrix by the number of synapses in the network. For the isotropic anisotropicity matrix, this scaling yields

$$\boldsymbol{\mathcal{A}}_0 = \boldsymbol{J}_{\pm}. \tag{3.7}$$

The isotropic weight space traversal implied by $\boldsymbol{\mathcal{A}}_0$ is equal in distance for all synaptic weights, with only the direction of the change varying among the weights. Though this weight modification policy is sufficient to train classification networks, there is relatively little diversity of ways in which the synaptic weights can change. A natural extension of $\boldsymbol{\mathcal{A}}$ which allows for greater diversity of synaptic weight modification, is to allow the portion of the total weight matrix change contributed by each synaptic weight to be randomly chosen. We define the anisotropicity matrix which implements this extension as

$$\boldsymbol{\mathcal{A}}_U = \frac{\boldsymbol{U}}{D/2} \tag{3.8}$$

where $\boldsymbol{U}$ is a matrix of dimensionality equivalent to $\boldsymbol{\mathcal{W}}$ for which each element is drawn from $U[0,1]$. The denominator ensures statistical compliance with the restriction that the elements of any matrix used as a realization of $\boldsymbol{\mathcal{A}}$ sum to unity. This constrain is very likely to be satisfied by $\boldsymbol{\mathcal{A}}_U$ for large networks because $\langle U[0,1]\rangle = 1/2$. Therefore, because

there are D non-zero elements in $\boldsymbol{U}$ the expectation value of the sum of the elements in $\boldsymbol{U}$ is $D/2$. Applying the same dimensional scaling as in Eq. **??** and simplifying, we find that

$$\boldsymbol{\mathcal{A}}_U = 2\boldsymbol{U}. \tag{3.9}$$

Using $\boldsymbol{\mathcal{A}}_U$ yields significantly more diversity of synaptic weight perturbations, but there is still a large class of possible perturbations which it is very unlikely to generate. These perturbations are those in which a large portion of the synaptic weights are left unchanged while some subset is changed significantly. In order to create a realization of $\boldsymbol{\mathcal{A}}$ which generates this type of perturbation we introduce a new stochastic control parameter, κ, which specifies the concentration of the total anisotropicity and is supported on the range $(0, 1/4)$. Thus, the anisotropicity of the matrix varies with the value of κ. We define the variable anisotropicity matrix to be

$$\boldsymbol{\mathcal{A}}_V = 2^{\frac{1}{1-\kappa}} \cdot power\left(\boldsymbol{U}, \frac{1}{1-\kappa}\right). \tag{3.10}$$

where the *power* function is the element-wise scalar exponentiation function. This definition is already scaled and simplified, as in Eq. **??**. It is clear that in the $\kappa \to 0$ limit $\boldsymbol{\mathcal{A}}_U$ is recovered. In the $\kappa \to 1/4$ limit, the matrix has only a few large elements, with the rest set nearly to zero. The upper bound of κ was determined empirically to be the largest value that does not result in a matrix which violates any of the constraints.

IV. Results

V. Conclusion

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. 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.

4. Block, H. D. "The Perceptron: A Model for Brain Functioning," *Reviews of Modern Physics*, *34*:123–135 (1962).

5. 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).

6. Bryson, A. E. and Y. C. Ho. *Applied Optimal Control*. New York: Blaisdell, 1969.

7. Corana, A., et al. "Minimizing Multimodal Functions of Continuous Variables with the &Ldquo;Simulated Annealing&Rdquo; algorithmCorrigenda for This Article is Available Here," *ACM Trans. Math. Softw.*, *13*(3):262–280 (September 1987).

8. Cybenko, G. "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, *2*(4):303–314 (1989).

9. Das, Arnab, et al. "Quantum annealing in a kinetically constrained system," *Phys. Rev. E*, *72*:026701 (Aug 2005).

10. Goffe, William L., et al. "Global Optimization of Statistical Functions with Simulated Annealing," *Journal of Econometrics*, *60*:65–99 (1994).

11. Haykin, Simon. *Neural networks: a comprehensive foundation*. Upper Saddle River, N.J: Prentice Hall, 1999.

12. Hebb, D. O. *The organization of behavior; a neuropsychological theory, (by) D.O. Hebb. Science Editions*. New York: John Wiley and Sons, 1967.

13. 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).

14. James, W. *The Principles of Psychology*. Number v. 1 in American science series: Advanced course, H. Holt, 1890.

15. Kirkpatrick, S., et al. "Optimization by simulated annealing," *SCIENCE*, *220*(4598):671–680 (1983).

16. Kohonen, Teuvo. "Correlation Matrix Memories," *IEEE Trans. Comput.*, *21*(4):353–359 (April 1972).

17. Lecchini-Visintini, A., et al. "Simulated Annealing: Rigorous finite-time guarantees for optimization on continuous domains," *ArXiv e-prints* (September 2007).

18. Lecun, Y. "Une procédure d'apprentissage pour réseau à seuil asymétrique," *Proceedings of Cognitiva 85, Paris*, 599–604 (1985).

19. 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.

20. Metropolis, N., et al. "Equation of State Calculations by Fast Computing Machines," *21*:1087–1092 (June 1953).

21. Minsky, M. and S. Papert. *Perceptrons*. Cambridge, MA: MIT Press, 1969.

22. Mukherjee, S. and B. K. Chakrabarti. "Multivariable optimization: Quantum annealing and computation," *European Physical Journal Special Topics*, *224*:17 (February 2015).

23. 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.

24. Piccinini, Gualtiero. "Computational Explanation in Neuroscience," *Synthese*, *153*(3):343–353 (2006).

25. 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).

26. Rosenblatt, F. "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, *65*(6):386–408 (1958).

27. Rumelhart, D.E., et al. *Learning Internal Representations by Error Propagation*. 1986.

28. Szu, Harold and Ralph Hartley. "Fast simulated annealing," *Physics Letters A*, *122*(3-4):157–162 (June 1987).

29. Tsallis, Constantino and Daniel A. Stariolo. "Generalized simulated annealing," *Physica A: Statistical and Theoretical Physics*, *233*(1-2):395–406 (November 1996).

30. von der Malsburg, Chr. "Self-Organization of Orientation Sensitive Cells in the Striate Cortex,"

31. Werbos, P. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD dissertation, Harvard University, 1974.

32. 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