Random Structure of Error Surfaces: Toward New Stochastic Learning Methods

Invited Presentation

Andrew B. Kahng
UCLA Department of Computer Science
Los Angeles, California 90024-1596

1 Introduction

Learning in neural networks can be formulated as global optimization of a multimodal error function that is defined over the high-dimensional space of connection weights. This global optimization is both theoretically intractable [26] [35] and difficult in practice. Traditional learning heuristics, e.g., back-propagation [31] or Boltzmann learning [11], are largely based on gradient methods or stochastic hill-climbing, reflecting traditional global optimization approaches (see, e.g., [10] for a survey). While these methods have shown promise on smaller problem instances, optimizing larger connectionist architectures raises several difficulties which motivate the present work:

- **No performance bounds.** Local optima can be arbitrarily far from global optima. Indeed, for certain "hard" combinatorial optimization instances, the expected result of either the steepest-descent or simulated annealing approaches will be no better than a random solution [5].

- **No estimates of minimal network topology.** We cannot tell *a priori* whether an n-node topology is capable of a prescribed discrimination task, nor do we know whether our training algorithm will in fact allow a network to achieve its capability. Thus, choice of network architecture is largely an *ad hoc* decision. For large problems, we can only implicitly justify trained solutions as the result of huge amounts of computation, or by arguing that the training methods had previously been effective on smaller problems.

- **Instability.** Training heuristics are unpredictable, and sensitive to initial random seeds [17]. Thus, we have little idea of the incremental benefit of spending additional CPU time on network training. Ideally, methods will be stable and exhibit a smooth tradeoff between performance and CPU cost.

- **Poor scaling.** Theoretical results on learning methods are essentially statements of convergence to local minima in the error surface. This is of little practical impact since local search heuristics exhibit an "error catastrophe" [15] as problems grow large. We require methods where solution quality scales with problem size.

- **Finite time requirements.** Finally, several learning algorithms (particularly Boltzmann-style algorithms) are "provably good", but only in the limit of infinite time. In practice, we require a theory which explicitly addresses the tradeoff between

This paper gives an overview of current work which is directed toward verifying, and exploiting in practice, a recent scaling model for neural network error surfaces. We begin the next section by reviewing a model which describes Boltzmann learning as stochastic search in the error surface [12] [13] [14]. The discussion also reviews a potentially far-reaching fractal model of neural network error surfaces as instances of a class of high-dimensional fractional Brownian motions (fBm). The main body of the paper then describes a series of experimental results for object classification via noisy sensor data in a mine detection application.

 0-8194-0875-1/92/$4.00

2 Review of Previous Work

In neural network learning, the usual strategy for searching for an optimum weight assignment is to generate a slight perturbation of the current solution, then decide whether this change should be accepted. How the perturbation is generated yields structure: certain elements of the solution space are directly *reachable* from others, and the induced reachability graph is called a *neighborhood structure* [22]. The quality of solutions defines a cost surface over the neighborhood structure, and optimization is search for a global optimum in this cost surface.

Existing classes of heuristics correspond to various rules for generating a new candidate solution and deciding whether to adopt the solution. For this discussion, we adopt the general stochastic hill-climbing template in Figure 1 known as *simulated annealing* or *Boltzmann learning* (BL) with temperature parameter T [16]. In Figure 1, $f(x)$ denotes the value of the cost function for the solution x.

Generate candidate solution x'
If $f(x') < f(x)$ then $x = x'$
/*always accept improvement */
Else $x = x'$ with probability $e^{-(f(x')-f(x))/T}$
/* Boltzmann acceptance */
otherwise $x = x$
/* leave x unchanged */

Figure 1: The Boltzmann Learning (BL) Inner Loop.

Basic optimization strategies, e.g., greedy and random search, are limiting cases of BL. Random search can be approximated by taking a *random walk* in the neighborhood structure, i.e., iteratively moving to a randomly chosen neighbor of the current solution, then returning the best cost value found on the walk. This is equivalent to BL with $T = \infty$. At the other extreme, BL with $T = 0$ is equivalent to greedy search. Practical cost surfaces seem to be best searched using strategies "intermediate" between randomness and greed.

To devise new and effective hill-climbing heuristics, we require a model for the smoothness or correlatedness of the cost surface. Certainly, for pathological surfaces with a single "gopher hole" as global minimum will require exhaustive search, and BL will actually be less efficient than enumeration or random search. On the other hand, BL will be no more useful than greed for "simple" surfaces. As noted in, e.g., [2] and [36], it seems that hill-climbing would be a win, as it has been in practical experience, when the error surface is "smooth, but not too smooth". Slices through small neural network error surfaces have been exhaustively plotted to gain insight into the structure of actual problems, but no general structural models have resulted.

With this in mind, previous work [12] [13] [14] has proposed a general scaling model for neural network error surfaces; we note that Sorkin [33] has investigated similar ideas for combinatorial optimization problems in VLSI layout. Useful definitions (see [12] [33] [38] for necessary background in fractal sets and random processes) are as follows.

Definition: A *similarity transformation*, defined by a scalar $r \in R$, maps $x \in R^n$ to rx.

Definition: An *affinity transformation*, defined by a vector $r \in R^n$, maps $x \in R^n$ to the vector $(r_1x_1, \ldots, r_nx_n)$.

Definition: A set $S \subseteq R^n$ is *self-similar* (resp. *self-affine*) if it is the disjoint union of N rotated and translated copies of rS, where rS is a similarity (resp. affinity) transformation.

Definition: A *fractional Brownian motion* (fBm) with parameter $H \geq 0$ is a sequence of random variables $X(t)$ such that the random variable $X(t_2) - X(t_1)$ is Gaussian with mean zero and variance

$$E[|X(t_2) - X(t_1)|^2] \propto |t_2 - t_1|^{2H} \tag{1}$$

Because any sample of a fBm is statistically similar to the original fBm [38], fractional Brownian motions are

called *statistical fractals*. Equation (1) implies that the scaling of fBm is power-law, e.g., at time t a "drunkard's walk" on the one-dimensional lattice has expected divergence of $t^{1/2}$ from the origin.

The *cost profile* over a random walk to be the sequence of cost values encountered as we iteratively move to random adjacent solutions. A fundamental observation due to Sorkin [33] is the following:

Fact 1: The cost profile over a random walk in a high-dimension fBm error surface will itself be a fBm in one dimension. □

In previous work reported in [13] [14], we have found strong evidence for the model of neural network error surfaces as high-dimensional fBm. Specifically, the parameter of fBm can be extracted by spectral analysis of the error profile as we take a random walk in weight space. A natural algorithmic approach to quantifying the *pseudo-fractalness* (i.e., smoothness) of the error surface is: (i) take a small (random-walk) sample of the weight space, (ii) treat the error values over this sample as a time series $X_0, \ldots, X_T$, (iii) take the discrete Fourier transform (DFT) $X(f,T)$

$$X(f,T) = 1/T \int_0^T X(t) e^{2\pi i f t} dt ,$$

and (iv) plot the resulting spectral density, or periodogram $S_X(f)$

$$S_X(f) = \lim_{T \to \infty} T \cdot |X(f,T)|^2$$

on a log-log scale to confirm the power-law scaling via a straight-line fit. For example, the 1-D drunkard's walk would correspond to fBm with parameter $H = 1/2$, and the fitted slope of the log-log plot of the power spectrum is $-1/2$.[1] Since the DFT algorithm is $O(n \ log \ n)$, where n is the length of the random walk, and since the sampling of error values along the random walk may be performed in parallel, we have a very efficient computational methodology.[2]

Over the past year and half, a number of experiments have addressed aspects of this general scaling model for large-scale optimization, dealing with a wide range of benchmark classification tasks from the literature, as well as practical tasks involving various types of real-world sensor data (e.g., ground-penetrating radar, FLIR images). We have implemented standard DFT code [28] along with standard moving-window smoothing techniques using C in a Sun-4 environment. Our results confirm that error surfaces over a wide range of classification tasks and connection topologies are indeed describable as high-dimensional fBm. We emphasize that the power-law scaling relationship is very strong in *all* of the examples that we have considered.

The most far-reaching implications of our scaling model have to do with bounded-time search, i.e., how should network training be performed given a limit of k time steps? Mitra et al. posed this problem in [25], but centered on imperfect convergence to stationarity in the general simulated annealing process. Under our model, we may address the special problem of searching a fractal surface for a global minimum; our analysis begins with the case where only a single value of T in the annealing inner loop is allowed (such a search is called a *diffusion* at temperature T). One can show:

Fact 2: In searching a high-dimensional fBm surface for a global optimum, the optimal diffusion temperature T (i.e., the value of T which gives the best expected result at time k) is a function of the k and the fBm parameter H. □

Sorkin [33] showed that the Boltzmann learning algorithm is provably good in the sense that it will reach the optimum solution in polynomial time on a class of deterministically constructed fractal surfaces. Moreover, this result can be slightly strengthened. This lends hope that Boltzmann learning, which is no more efficient than exhaustive search on pathological error surfaces, is actually very effective for "real" error surfaces.

[1] In general, the value of H is estimated by -2 times the slope of the log-log plot of the amplitude of the Fourier transform plotted against time in the random walk.

[2] As a minor technical point, note that it is theoretically impossible to *prove* that a finite sample of a cost profile is truly fBm; however, we may subject the sample to statistical tests. We formalize this distinction by defining an error surface to be *pseudo-fractal* if the cost profile over a random walk in the neighborhood structure is statistically indistinguishable from fBm.

Experiments in [13] plotted average best-so-far error (BSFE) (the minimum error seen during the first k steps of the random walk) versus T, and obtained unimodal curves which confirm both the existence of an optimal T each given k, as well as the expected shift of the optimum temperature T as the time bound k varies. Studying the performance of bounded-time hill-climbing search in this way suggests the concept of a natural time scale, or relaxation time, for each optimization instance [32]. Thus, the scaling of best-so-far energy can be used to predict ground state energy in the annealing, i.e., the cost of the global optimum solution. Other results in [12] [13] have addressed the robustness of the random walk-based estimates.

As noted in [14], the most promising result of the scaling model is a methodology for applying the BSFE scaling in Boltzmann learning to determine connectionist architectures that are *likely* to be trainable to prescribed capability within a given time bound. In other words, we can examine the early progress of the Boltzmann learning algorithm to see if success is likely; if not, then a new connectionist topology can be tried. The remainder of this paper, we report on some recent experiments to this end.

3 Experimental Results With Noisy Sensor Data

Previous work reported in [13] and [14] observed strong straight-line fits have been observed in neural network error surfaces. However, these results were obtained in the context of such "classic" problems as the parity or encoder functions. With these tasks, there is a high degree of symmetry in the problem structure and thus the relevance of the fractal model to practical applications remained unclear. With this in mind, our recent work has studied the scaling properties of neural network error surfaces induced from ground sensor data obtained via the wave guide beyond cutoff (WGBCO) microwave technique [29] at U.S. Army Belvoir RD&E Center.

The preliminary experiments that we report shed light such issues as (i) the optimal learning strategy given k time steps, and (ii) determination of the simplest neural network architecture that can be *expected* to be trained to a desired level of performance in k steps. Our testbed has been previously treated by researchers from Colorado State University and Stanford University, and so the reader is also referred to such references as [34] and [21].

3.1 Description of Datasets

The data used in this study was collected by the U.S. Army at the Fort Belvoir research center. The data was taken from two virtual mine lanes which respectively contained nylon and wood target blocks buried in dry loamy soil. The first mine lane (60 feet by 3 1/3 feet) contained 15 nylon target blocks and the second lane (63 feet by 3 1/3 feet) contained 12 wood target blocks; each block measured 12"x12"x3". Readings were collected at equally spaced points (every 1.25 inches) in the lanes, using a microwave separated-aperature sensor. Although data was collected for 51 different frequencies at regular intervals between 600 MHz and 1 GHz, the only frequency used in this analysis was 792 MHz, which was the resonant frequency of the sensor.

Given the mine lane dimensions listed above, the data files were of size 486x27 for the nylon target lane and 504x27 for the wood target lane. Two additional "ground truth" files were provided by colleagues at Belvoir RD&E Center, indicating whether a target was actually present at each of the data points in the two lanes. Because each target has size of approximately 11x11 data points but is allowed to take on any orientation, a window of size 15x15 is necessary to ensure that an entire target can be contained within one window. Therefore, square windows of this size are used in our study. Our experiments differ considerably from previous studies which used these datasets [21] [34]. To be specific, earlier work was concerned with adaptive learning rules, and addressed the training of very large network topologies containing hundreds of nodes. By contrast, we examine the rapidity with which various simple network architectures can be reliably trained; our goal is to determine the simplest architecture which high likelihood of being trained to achieve perfect target classification. (As will be seen below, very simple networks suffice to solve this particular target recognition problem, suggesting that the problem is in some sense an "easy" benchmark.)

Given the 15x15 size of each window, a naive neural network implementation using uncompressed input data would require 225 input nodes and a very large amount of computer time. Thus, we have used a the Karhunen-Loeve (KL) transform, a principal components method, to reduce the number of inputs while retaining much of the information from the original inputs. Previous workers have also applied the KL transform to the present image recognition task (for example, see [23], [34] and [37]). The KL transform requires calculation of a 225x225 covariance matrix over the input windows and then computation of eigenvectors for the covariance matrix. Each eigenvalue represents the portion of total variance of the 225 represented by the corresponding eigenvector.

In our analysis, each target lane is divided into two parts: the top half is for training the neural networks and the bottom half is for network simulation. The covariance matrix used by the KL transform was computed using all possible windows in the top halves of the two lanes (there are 2,977 windows in the top half of the nylon lane, and 3,094 windows in the top half of the wood lane). This large number of windows was used with the expectation that a larger sample for the covariance matrix will improve the accuracy of the compresed data, thus improving simulations in the bottom halves of the arget lanes. After computation of the eigenvectors and eigenvalues, the eigenvectors were sorted by the size of their eigenvalues. Each eigenvalue represents the portion of total variance of the window data as "captured" by its corresponding eigenvector. The results of calculation indicate that the first 10 eigenvectors contain 73.2 percent of the variance in the 792 MHz readings from the 225 window points. In other words, we can reduce the number of data points in each window from 225 to to 10 and retain about 70 percent of the original information (as measured by the covariance matrix). (It should be noted that a covariance matrix only represents dependencies between pairs of variables, and misses information if the dependencies between window points are more complicated.) Table 1 shows the first 15 eigenvalues.

Rank	Eigenvalue	% of Variance	Cumulative %
1	2,141.3	21.0	21.0
2	1,429.7	14.0	35.0
3	1,083.0	10.6	45.6
4	676.8	6.6	52.2
5	604.0	5.9	58.1
6	535.5	5.3	63.4
7	290.5	2.9	66.3
8	281.0	2.8	69.1
9	220.4	2.2	71.3
10	190.1	1.9	73.2
11	148.4	1.4	74.6
12	143.8	1.4	76.0
13	121.9	1.2	77.2
14	105.1	1.0	78.2
15	101.9	1.0	79.2

Table 1: Eigenvalues of the covariance matrix from all top half windows. Total variance = 10,186.1; average variance per window point = 10,186.1 / 225 = 45.27; average standard deviation per window point

Finally, a training set was chosen from windows in the top half of each of the two target lanes. To facilitate learning, windows were chosen that contain either all of a target block or no part of a target block.[3] Each wood or nylon target was represented in the training set by one target window. One window was also chosen in the space between any two consecutive targets to represent a background window. (In the target lanes, each target is positioned far enough away from other targets that at least a full background window can be placed between them. Furthermore, the targets are placed in the lanes so that no two targets can occupy space in the same row of data.) A total of 29 training windows were selected; these may be categorized as follows: nylon target (8), nylon background (8), wood target (6) and wood background (7).

[3] An attempt was also made to use the same training windows as in a previous study involving the same data [34]. However, exactly which training windows were used in [34] is not clear from the descriptions provided. Consequently, the windows in this study are at least slightly different.

3.2 Experiments With Neural Network Architectures

We studied feedforward networks with one hidden layer and connections existing only between adjacent layers. In this class of architectures, each node has an "offset" or "bias" weight in addition to weights between itself and all other nodes in the previous layer. The standard logistic function is used as a squashing function on the activation to all hidden and output nodes. Most of the network architectures we tested have two output nodes (alternate architectures with one output node and with three output nodes were also considered). In the two-output architectures, the desired outputs take on the following values: (i) 0 0 – a background window (nylon or wood lane); (ii) 1 0 – a nylon target window; or (iii) 0 1 – a wood target window.

The expected output and first six input values obtained from the Karhunen-Loeve transform are presented in Table 2 for each of the 29 training cases. From the table it is evident that the sign of the first input is almost sufficient for distinguishing between target and background windows, but not between nylon and wood targets.

Nylon Target	Wood Target	1st Input	2nd Input	3rd Input	4th Input	5th Input	6th Input
1	0	96.12	-5.17	29.50	-6.55	52.01	-3.98
1	0	102.86	7.80	-33.00	-24.20	25.80	16.49
1	0	58.40	-37.51	20.82	-17.03	101.04	45.64
1	0	8.67	44.06	-60.68	-32.36	34.13	15.39
1	0	33.27	-59.84	9.81	33.42	13.45	-14.33
1	0	98.26	-13.73	22.63	-24.66	43.33	47.20
1	0	83.24	32.29	36.56	-35.03	62.51	17.79
1	0	41.21	10.41	39.76	-41.83	27.80	55.83
0	0	43.70	7.63	-22.86	-6.97	2.30	-15.64
0	0	-8.10	28.77	3.40	-0.03	-27.23	5.66
0	0	-21.49	-0.93	-3.28	-38.03	10.62	-20.61
0	0	-2.10	40.82	12.56	-36.51	22.93	4.98
0	0	-12.53	-32.44	-2.35	10.69	-1.52	16.30
0	0	-50.47	-12.87	28.01	14.54	-21.59	25.10
0	0	-25.02	-2.41	-15.24	-5.69	-20.78	4.27
0	0	-14.02	31.78	-2.38	9.82	-18.35	15.16
0	1	51.58	-31.24	47.67	-43.51	49.83	17.51
0	1	54.44	-26.86	18.20	-44.45	43.62	32.80
0	1	8.93	-23.58	-1.36	-15.97	53.21	25.58
0	1	59.04	-25.32	53.49	-9.50	27.74	9.25
0	1	102.19	7.77	-44.13	-8.25	51.67	17.68
0	1	23.11	-5.98	-6.27	-1.37	17.06	33.42
0	0	-62.72	47.22	-3.31	-7.43	10.15	4.87
0	0	-49.96	10.22	-26.63	-14.10	3.90	-24.10
0	0	-40.75	21.53	41.51	-24.67	27.23	9.29
0	0	-67.49	-15.26	-53.75	19.59	5.34	6.13
0	0	-61.43	15.98	-2.67	-14.08	-10.00	14.56
0	0	-88.16	-39.40	-39.30	-20.47	50.03	16.08
0	0	-66.08	25.35	3.63	-25.55	7.84	9.18

Table 2: Training input.

The networks were trained using the BL algorithm with *constant* temperature T; the objective function for minimization was the sum of squared errors (SSE) between the expected and actual network outputs.[4] Initial weights were generated randomly from a uniform distribution between -0.1 and 0.1. Relative to final weights, these starting values were quite close to zero, again reflecting common practice. At each iteration, new weights were generated from the old weights using a randomly chosen perturbative step from the range -0.05 to +0.05.

The various network architectures were trained with different *fixed* temperatures (i.e., a degenerate annealing schedule) to compare the effectiveness of the Boltzmann learning algorithm under different conditions. The parameters whose variation we studied most in this study were: (i) number of input nodes; (ii) number of hidden nodes; (iii) the (fixed) temperature of the Boltzmann *diffusion*; and (iv) the number of iterations, or steps in the search process.

[4]This objective function is popular for the ease with which gradients may be computed. Separate work has made comparisons between BL and the gradient descent and conjugate gradient classes of training algorithms, but is beyond the scope of the present discussion.

The inputs used were the values for each window obtained by the the Karhunen-Loeve transform. The inputs chosen were always the i input values created by the eigenvectors with the i largest eigenvalues. In other words, the i inputs were used that maximize the amount of variance in the window covariance matrix. (Additionally, all of the original sensor values were normalized so that their means over the first half of the target lanes were equal to zero.)

The number of hidden nodes was varied from 2 to 12, with two additional architectures trained with 16 and 32 hidden nodes. Boltzmann temperatures were tested at values of 0.5, 0.05, 0.005, and 0. (Recall from the statement of the Boltzmann loop that at temperature zero, a random step is taken only if it produces lower objective function.) At each iteration number equal to a power of two, statistics were compiled on the best-so-far value of the objective function; each run lasted for a total of 65,536 iterations. For each network architecture and Boltzmann temperature, a total of ten training runs were made from different starting values. More runs may have improved the minimum objective values achieved, but were not attempted because of CPU constraints.

Based on previous theoretical and experimental results [13] [14], we expected the following results. First, we expected that a larger architecture would train more slowly than a smaller architecture, especially if both were capable of being trained to the same value of the objective function. In other words, given architectures with the same "trainability" attributes, the more complicated topologies would be harder to train. Second, we expected to find optimal temperatures that would vary for different architectures in some systematic way. Finally, we expected that this optimal temperature would almost always be greater than zero, since a temperature of zero prevents the algorithm from escaping from local optima.

3.3 Results and Conclusions

Tables 3, 4 and 5 give the average sum of squared errors after 4,096, 16,384 and 65,536 iterations.[5] These results, along with those of Table 6, suggest some interesting properties of training on the microwave sensor data:

1. The optimal Boltzmann temperature varies depending on the network architecture. In almost all cases, however, the a zero temperature is not optimal.

2. The trainability of networks (based on the average size of the objective function after a constant number of steps) increases as the the network size increases, except for very large architectures. For large architectures trainability decreases at a slow rate.

   It should be noted, however, that the actual number of calculations per step (based on the total number of weights) increases linearly in the number of input nodes times the number of hidden nodes. However, a fair comparison can be made, for instance, between a 4 input, 4 hidden node network after 64k steps and an 8 input, 8 hidden node network after 16k steps. The smaller network achieves a average objective function of 1.31 at temperature 0.05, while the larger one achieves a 0.17 average objective function. This implies that large architectures can train more easily than smaller ones using approximately the same number of computations. This observation cannot be explained by the inability of a 4 x 4 x 2 network to "learn" these training cases: As can be seen in Table 2, a 4 x 4 x 2 network can be trained to an objective function very close to zero.

   Another "fair" comparison can be made between architectures with 4 input nodes and 4, 8, 16, and 32 hidden nodes after 64k, 32k, 16k, and 8k steps respectively. The average best-so-far SSE in these cases were 1.31, 0.18, 0.26, and 0.61 at temperature 0.05. These values again indicate a general trend of increasing trainability for larger networks except for very large networks, where training gradually becomes more difficult with increasing size.

[5] Parallel experiments treated network architectures with three output nodes; each output node thus represented either a nylon target window, a wood target window, or a background window. The difference in trainability and in performance of simulations was not qualitatively different. A parallel study by Mark Plutowski of UC San Diego trained a network with just one output node (values of 0.2 for background windows, 0.5 for nylon target windows, and 0.8 for wood target windows). Again, the results were not qualitatively different.

3. Of the network architectures tested here, the 4 input and 4 hidden unit and the 8 input and 2 hidden unit networks appear to be the smallest networks that can be trained almost perfectly (for the training cases alone). Note that these are much smaller than those obtained by previous researchers who have worked with the same datasets.

Number of Inputs	Temp	Number of Hidden Nodes							
		2	4	6	8	10	12	16	32
2	0.5	6.43	5.40	5.18	4.42				
	0.05	6.26	4.28	2.50	1.84				
	0.005	5.86	4.30	3.83	2.43				
	0	6.11	5.45	4.91	3.17				
4	0. 5	6.30	4.84	3.18	3.02	2.88	2.69	2.16	1.82
	0.05	4.52	1.31	0.14	0.07	0.01	0.02	0.01	0.004
	0.005	4.84	2.32	0.75	0.82	0.60	0.57	0.81	1.10
	0	5.28	2.51	2.14	0.98	0.53	1.25	1.02	1.38
6	0. 5	6.43	4.60	4.14	3.11	2.74	3.56		
	0.05	4.29	1.58	0.21	0.17	0.02	0.02		
	0.005	4.77	1.58	0.22	0.20	0.003	0.003		
	0	5.00	2.55	1.38	0.11	0.45	0.21		
8	0.5	6.35	5.78	3.81	3.06			2.32	
	0.05	4.30	0.49	0.01	0.02			0.006	
	0.005	3.55	0.77	0.01	0.002			0.19	
	0	4.85	2.44	0.02	0.18			0.34	
10	0.5		4.86	4.17					
	0.05		0.52	0.17					
	0.005		0.84	0.001					
	0		2.05	1.05					
12	0.5		4.81	4.11					
	0.05		0.31	0.01					
	0.005		0.10	0.001					
	0		1.27	0.62					

Table 3: Average (of 10 runs) Best So Far SSE after 65,536 Steps. Step size used is between -0.05 and +0.05; SSE computed over 29 training cases.

Number of Inputs	Temp	Number of Hidden Nodes					
		2	4	6	8	10	12
2	0.5	6.93	5.84	5.58	5.29		
	0.05	6.39	5.10	3.00	2.45		
	0.005	5.99	5.07	4.27	3.61		
	0	6.17	5.47	5.25	4.67		
4	0.5	7.14	6.08	4.36	4.30	4.28	4.09
	0.05	5.02	2.15	0.96	0.48	0.29	0.35
	0.005	5.15	3.02	1.66	2.37	2.35	2.15
	0	5.33	3.15	2.99	2.27	2.43	2.37
6	0.5	6.91	6.28	5.11	4.48	4.49	4.41
	0.05	4.41	2.60	0.56	0.57	0.18	0.17
	0.005	5.11	2.19	0.80	1.20	1.20	1.05
	0	5.12	2.94	1.88	0.99	1.46	2.23
8	0.5	6.67	6.83	5.39	4.14		
	0.05	4.68	0.95	0.36	0.17		
	0.005	4.30	1.76	0.36	0.38		
	0	5.56	2.62	0.53	0.81		
10	0.5		6.00	5.84			
	0.05		1.07	0.44			
	0.005		1.93	0.23			
	0		2.23	1.23			
12	0.5		5.90	5.35			
	0.05		0.57	0.28			
	0.005		0.80	0.13			
	0		1.39	0.78			

Table 4: Average (of 10 runs) Best So Far SSE after 16,384 Steps. Step size used is between -0.05 and +0.05; SSE computed over 29 training cases.

Results were ambiguous regarding the hypothesis that the surface of the search space in the neural network optimization problem is pseudo-fractal. Recall that a primary motivation for our work is that an approximation

Number of Inputs	Temp	Number of Hidden Nodes					
		2	4	6	8	10	12
2	0.5	8.08	6.63	6.50	6.40		
	0.05	6.71	5.44	4.53	4.26		
	0.005	6.19	5.38	5.17	4.93		
	0	6.27	5.57	5.45	5.26		
4	0. 5	8.39	7.10	6.01	5.93	5.91	5.73
	0.05	5.96	3.66	2.88	2.24	2.10	1.88
	0.005	5.34	4.28	2.99	4.25	4.17	3.43
	0	5.85	4.52	4.06	3.71	4.31	3.80
6	0. 5	8.04	7.56	6.69	6.43	5.81	5.46
	0.05	5.62	3.83	2.12	2.17	1.80	1.73
	0.005	6.11	3.14	2.89	3.31	3.66	3.23
	0	5.23	3.91	3.26	3.37	3.27	3.85
8	0.5	7.77	7.34	7.17	6.12		
	0.05	5.35	2.71	2.00	1.41		
	0.005	5.30	2.73	2.30	4.33		
	0	5.82	3.46	2.58	3.07		
10	0.5		7.78	7.05			
	0.05		2.25	1.65			
	0.005		3.09	1.88			
	0		3.04	2.47			
12	0.5		7.09	6.65			
	0.05		1.85	1.23			
	0.005		2.19	1.77			
	0		2.12	2.01			

Table 5: Average (of 10 runs) Best So Far SSE after 4,096 steps. Step size used is between -0.05 and +0.05; SSE computed over 29 training cases.

Number of Inputs	Number of Hidden Nodes					
	2	4	6	8	10	12
2	5.23	2.38	1.37	1.02		
4	3.26	0.09	0.02	0.000	0.001	0.003
6	3.22	0.01	0.001	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000		
10		0.000	0.000			
12		0.000	0.000			

Table 6: Minimum (over 10 runs and over all temperatures) of Best So Far SSE after 65,536 Steps. Step size used is between -0.05 and +0.05; SSE computed over 29 training cases.

of the fractal dimension of the random walks may be used as a predictor of which architectures can be trained to an error of zero, which architectures will train most quickly, or perhaps which Boltzmann diffusion temperature will be optimal.

We estimated the fBm parameter for various network architectures using random walks with 16,384 steps. To apply the methodology of [13], it was necessary to divide the random steps into discrete sizes: the change in each weight was allowed to take on 21 possible values uniformly between -0.05 and 0.05 (including 0). The DFT was performed on each random walk, and resulting magnitudes were smoothed using a moving-window geometric average with window size 20. To increase the accuracy of estimation for the parameter of fBm, the results of 10 separate random walks for each network architecture were averaged. An estimate of the standard error of the average estimate was also computed. Table 7 summarizes the estimates of "fractal dimensions".

Clearly, there are only small variations in the slope of the straight-line fit (i.e., the parameter of fBm). We may test the significance of these variations as follows. If we assume that the parameter estimates for each architecture are normally distributed, we may apply a two-sample Student's t test to test the hypothesis that any two values in Table 7 are the same. The t statistic to test whether two sample means come from the same distribution is

Number of Inputs	Number of Hidden Nodes				
	2	4	6	8	10
2	.4239	.4130	.4150	.4083	.4060
	(.0027)	(.0055)	(.0042)	(.0054)	(.0034)
4	.3970	.4068	.4037	.3977	
	(.0046)	(.0047)	(.0046)	(.0044)	
6	.4184	.3927	.3997	.3917	.3872
	(.0032)	(.0053)	(.0030)	(.0044)	(.0049)
8	.4203	.4008	.3916	.3938	
	(.0045)	(.0059)	(.0043)	(.0046)	
10	.4181	.4004	.3851	.3884	
	(.0033)	(.0062)	(.0061)	(.0034)	
12			.4008		.3867
			(.0022)		(.0031)

Table 7: Estimated fractional Brownian motion parameters of different architectures using random walks of length 16,384; average taken over 10 random walks. (Standard Errors in parentheses.)

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{1}{n_1} + \frac{1}{n_2}} \left[\frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2} \right]^{1/2}}$$

where $\overline{X}_1$ and $\overline{X}_2$ are the sample means; n_1 and n_2 are the sizes of the samples; and S_1^2 and S_1^2 are the sample variances. This statistic is Student distributed with $n_1 + n_2 - 2$ degrees of freedom. For example, to test the Brownian motion parameter estimate for the 2 input and 2 hidden node network versus that for the 4 input 2 hidden node network, we substitute $\overline{X}_1 = .4239$; $\overline{X}_2 = .3970$; $n_1 = n_2 = 10$; $S_1^2 = 9 * .0027^2 = 6.56 * 10^{-5}$; and $S_2^2 = 9 * .0046^2 = 1.90 * 10^{-4}$. The t-value is then 8.85, which is significant at the 95 and 99 percent levels. (The thresholds for 18 degrees of freedom are 2.101 and 2.878 for the 95 and 99 percent confidence levels, respectively.) A statistical test of the hypothesis that all the Brownian parameters are the same would require an analysis of variance using all of the slopes used to create Table 7. This test has not yet been made.

Table 7 shows only a small variation in the fBm parameter for different architectures, and no clear relation relation between the size of the network and the "fractal dimension" of its cost surface is discernible. With this in mind, current work is more closely investigating several possibilities, including: (i) that the error surface is not fractal; (ii) that our measure of the "fractal dimension" of the cost surface is not correct; and (iii) that the "fractal dimension" does not vary significantly for different neural networks which are being trained on the same data.

While these experimental results are not as supportive as earlier data, the new directions in the selection and training of neural networks suggested by the theoretical scaling model of [13] [14] are still promising. In particular, the implications of the model for bounded-time training seem practically useful; thus, current work continues to pursue methodologies which effectively match the network architectures with the intrinsic difficulty of the recognition task.

4 Acknowledgements

The author would like thank Don Franklin, Charles Amazeen and David Poole, as well as many other individuals at U.S. Army Belvoir RD&E Center, for many interesting discussions. At UCLA, Kenneth D. Boese was instrumental in obtaining the experimental results described above.

References

[1] E. H. L. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, New York, Wiley, 1989.

[2] P. Baldi, "Linear Learning: Landscapes and Algorithms", *Proc. Neural Information Processing Systems*, 1988, pp. 65-72.

[3] E. B. Baum and D. Haussler, "What Size Net Gives Valid Generalization?", *Neural Computation* 1 (1), 1989.

[4] E. O. Brigham, *The Fast Fourier Transform*, Prentice-Hall, 1974.

[5] T. N. Bui, S. Chaudhuri, M. Sipser and F. T. Leighton, "Graph Bisection Algorithms With Good Average-Case Behavior", *Combinatorica* 7(2) (1987), pp. 171-191.

[6] B. Dubuc, J. F. Quiniou, C. Roques-Carmes, C. Tricotand S. W. Zucker, "Evaluating the Fractal Dimension of Profiles",*Physical Review A* 39 (3) (1989), pp. 1500-1512.

[7] B. Dubuc, S. W. Zucker, C. Tricot, J. F. Quiniouand D. Wehbi, "Evaluating the Fractal Dimension of Surfaces",*Proc. R. Soc. Long. A* 425 (1989), pp. 113-127.

[8] G. A. Edgar, *Measure, Topology, and Fractal Geometry*, New York, Springer-Verlag, 1990.

[9] K. J. Falconer, *The Geometry of Fractal Sets*, Cambridge University Press, 1985.

[10] R. Hecht-Neilsen, *Neurocomputing*, Addison-Wesley, 1990.

[11] G. E. Hinton, "Deterministic Boltzmann Learning Performs Steepest Descent in Weight Space", *technical report* CRG-TR-89-1, Univ. of Toronto, 1989.

[12] A. B. Kahng, "Studies of Architectures and Algorithms for Object Recognition in Difficult Environments", ARO SFRP DAAL03-86-D-0001, 1990.

[13] A. B. Kahng and G. Robins, "On Structure and Randomness in Practical Optimization", *UCLA Computer Science Annual: 1991*, University of California at Los Angeles, 1991.

[14] A. B. Kahng, "Exploiting Fractalness in Error Surfaces: New Methods for Neural Network Learning", to appear in *Proc. IEEE Intl. Symp. on Circuits and Systems*, San Diego, June 1992.

[15] S. Kauffman and S. Levin, "Toward a General Theory of Adaptive Walks on Rugged Landscapes", *J. Theoretical Biology* 128 (1987), pp. 11-45.

[16] S. Kirkpatrick, C. D. Gelatt, Jr. and M.P. Vecchi, "Optimization by Simulated Annealing", *Science* 220(4598) (1983), pp. 671-680.

[17] J. F. Kolen and J. B. Pollack, "Back Propagation is Sensitive to Initial Conditions", *technical report* TR 90-JK-BPSIC, Ohio State Univ., 1990.

[18] T.W. Korner, *Fourier Analysis*, Cambridge University Press, 1988.

[19] M. Krentel, "Structure in Locally Optimal Solutions", *Proc. IEEE Symp. on Foundations of Computer Science* , 1989, pp. 216-221.

[20] H. Lass and P. Gottlieb, *Probability and Statistics*, Addison-Wesley, 1971.

[21] M. A. Lehr, "Overview of Work Done for Quarter Ending 9/30/90", Stanford University Department of Electrical Engineering.

[22] D. C. Llewellyn, C. Tovey and M. Trick, "Local Optimization on Graphs", *Disc. Applied Math.* 23 (1989), pp. 157-178.

[23] H. A. Malki and A. Moghaddamjoo, "Using the Karhunen-Loeve Transformation in the Back-Propagation Training Algorithm", *IEEE Trans. on Neural Networks* 2:1 (January 1991), pp. 162-165.

[24] B. Mandelbrot, *The Fractal Geometry of Nature* , New York, W. H. Freeman, 1982.

[25] D. Mitra, F. Romeo and A. Sangiovanni-Vincentelli, "Convergence and Finite-Time Behavior of Simulated Annealing", *Adv. in Applied Probability* 18(1986), pp. 747-771.

[26] P. M. Pardalos and G. Schnitger, "Checking Local Optimality in Constrained Quadratic Programming is NP-Hard", *Operations Research Letters* 7(1) (1988), pp. 33-35.

[27] H. O. Peitgen and D. Saupe, eds., *The Science of Fractal Images* , Springer-Verlag, 1988.

[28] W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge, Cambridge University Press, 1988.

[29] L. S. Riggs and C. A. Amazeen, "Research Efforts with the Waveguide Beyond Cutoff or Separated Aperture Dielectric Anomaly Detection Scheme", *report*, Belvoir RD&E Center, December 1989.

[30] F. Romeo and A. Sangiovanni-Vincentelli, "Probabilistic Hill Climbing Algorithms: Properties and Applications", *Proc. Chapel Hill Conf. on VLSI* , 1985, pp. 393-417.

[31] D. E. Rumelhart, J. L. McClelland et al., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* , Cambridge, MIT Press, 1986.

[32] P. Sibani, J. M. Pedersen, K. H. Horrmann and P. Salamon, "Monte Carlo Dynamics of Optimization Problems: A Scaling Description", *draft* , May 1990.

[33] G. B. Sorkin, "Simulated Annealing on Fractals: Theoretical Analysis and Relevance for Combinatorial Optimization", *Proc. Sixth MIT Conf. on Adv. Research in VLSI* , March 1990, pp. 331-351.

[34] S. A. Stricker, M. R. Azimi-Sadjadi, and D. E. Poole, "Application of the Karhunen-Loeve Transform for Target Detection and Classification Using Neural Networks", paper in progress.

[35] A. Torn and A. Zilinskas, *Global Optimization* , Lecture Notes in Computer Science 350, G. Goos and J. Hartmanis,eds., Springer-Verlag, 1987.

[36] D. Touretzky, "Analyzing the Energy Landscapes of Distributed Winner-Take-All Networks", *Advances in Neural Information Processing Systems I*, D. Touretzky, ed., Morgan Kaufmann, 1989, pp. 626-633.

[37] L. Torres-Urgell and R. L. Kilrlin, "Adaptive Image Compression Using Karhunen-Loeve Transform", *Signal Processing* 21 (1990), pp. 303-313.

[38] R. F. Voss, "Fractals in Nature: From Characterization to Simulation", in *The Science of Fractal Images* (H. O. Peitgen and D. Saupe, eds.), New York, Springer-Verlag, 1988.