01 Jan 1991

An Empirical Analysis of Backpropagation Error Surface Initiation for Injection Molding Process Control

Alice E. Smith

Elaine R. Raterman

Cihan H. Dagli
Missouri University of Science and Technology, dagli@mst.edu

Recommended Citation

A. E. Smith et al., "An Empirical Analysis of Backpropagation Error Surface Initiation for Injection Molding Process Control," *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics, 1991. 'Decision Aiding for Complex Systems'*, Institute of Electrical and Electronics Engineers (IEEE), Jan 1991.
The definitive version is available at https://doi.org/10.1109/ICSMC.1991.169905

An Empirical Analysis of Backpropagation Error Surface Initiation for Injection Molding Process Control

Alice E. Smith
Department of Industrial Engineering
University of Pittsburgh
1031 Benedum Hall
Pittsburgh, PA 15261

Cihan H. Dagli
and Elaine R. Raterman
Department of Engineering Management
University of Missouri - Rolla
Rolla, MO 65401

Abstract - **Backpropagation neural networks train by adjusting initially random interconnecting weights according to the steepest local error surface gradient. This paper examines the practical implications of the arbitrary starting point on the error landscape on the ensuing trained network. The effects on network convergence and performance are tested empirically, varying parameters such as network size, training rate, transfer function and data representation. The data used is live process control data from an injection molding plant.**

I. INTRODUCTION

Multi-layer perceptrons with non-linear transfer functions trained by backpropagation are the most popular form of neural networks today. They are applicable to a wide range of problems and are readily available in software simulation. The backpropagation training algorithm, fully described in Rumelhart *et al.*, works by adjusting the weights connecting neurons to move in the direction of greatest error reduction [4]. The network after training should consist of weights which are optimized to produce the smallest output error possible over the training set.

Theoretically the weight change should be done in infinitely small increments after the presentation of the complete training set. In real applications, weights are normally adjusted after the presentation of each input vector and move along the error surface in steps of fixed or variable size. The starting point on the error landscape is almost always arbitrary, with all variable weights set to random values between -1 and +1.

There are several noted problems with backpropagation which stem from this arbitrary initial weight state and the method of updating weights. First, since backpropagation only moves downward on the error surface, flat or upward sloping areas result in entrapment. Second, depending on the starting point on the error surface, a network may or may not encounter these kinds of areas during training. Third, the step size down the error surface gradient is not optimized, and may in fact cause the network to overshoot the absolute minimum of the error topology.

These problems are generally known, yet most applications of backpropagation do not address them explicitly. The purpose of the research presented in this paper is to determine empirically whether these relatively ignored problems have a significant effect on the training and operation of most networks. A real process control data set is used as the basis for this pattern classification task.

II. BACKPROPAGATION AND ITS ERROR SURFACE

A. Characteristics of the Error Surface

Backpropagation works by calculating an output error during training, which is the difference between the actual output and the desired (teacher) output. Weights are adjusted in proportion to the gradient (slope) at that point on the error surface. The training rate, n, along with the gradient determine the actual step size along the error landscape, as shown in Fig. 1. Steep areas will result in large steps while flat areas result in small steps in respect to the direction of decreasing error. In an ideal situation, the training rate would increase in flat areas to encourage

ISSN# 0-7803-0233-8/91 $1.00

movement across such regions, but be smaller in steep areas so overshoot cannot occur.

However, in practice the error topology of a backpropagation network to be trained is largely unknown. Previous research has indicated that in general the error landscape consists of flat areas and troughs with little slope, impairing the search of the best direction in which to move the weights [2]. For networks with linear transfer functions only, it has been proven that no local minima exist, although saddle points may be present [1]. The more commonly used networks with non-linear, usually sigmoidal, transfer functions can encounter local minima [2]. Both local minima and flat areas can cause a network to settle on weights which are non-optimal, as shown in Fig. 2.

Due to the nature of the error landscape, even nearing an absolute minimum (there are many because of the many different weight combinations available to a network) may not guarantee success. The training rate, not infinitesimally small, may cause an overshoot as shown in Fig. 2. Reducing the rate unnecessarily, however, will create inefficient training.

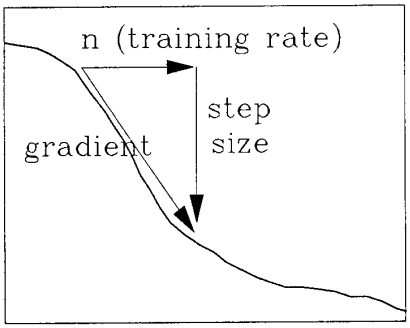


Fig. 1. Components of movement along the error surface during backpropagation training.

B. *Improving Gradient Descent*

Research has attacked several avenues to improve gradient descent. One method has been to alter the descent method itself. Besides the generally adopted momentum or smoothing term [7], which takes into account past weight changes on present ones, an acceleration term from the second derivative has speeded descent when a minimum is near [6]. Altering the sigmoid transfer function to ensure that values near 0 and 1 do not occur has also speeded training by eliminating extremely small weight updates [11]. Since the descent algorithm is designed for descent only, escape from local minima could be a desirable property. This can been accomplished through stochastic techniques which allow movement in directions other than the steepest descent [3, 10].

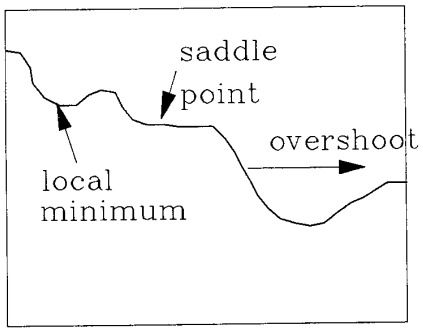


Fig. 2. A two dimensional interpretation of problems with error surface descent.

The training rate (n), since it influences step size, is also subject to improvement. As mentioned earlier, a static step size can cause both inefficiency and non-optimization. Recalculating n during each training cycle ensures that overshoot will not occur [8], but complicates the search.

Finally, perhaps the most desired improvement would be to select initial weights so that they facilitate training. This approach has been done with a Gaussian transfer function where a portion of the training set is first characterized by cluster analysis [9]. Another paper describes how optimum weights can be found using a different error function for each layer based on errors output in subsequent layers [5].

III. RESEARCH OF ERROR SURFACE INITIATION SENSITIVITY

A. *The Process Control Problem*

The application chosen for empirical analysis is process control of an injection molding operation. The manufacturer makes various diameters and lengths of brake linings. The data used consists of 88 samples of one product over a period of six

months. The process parameters measured included 16 independent variables, of which 13 were used for analysis. The output variable of interest was the mean outer diameter of the product. This application involves training a network to recognize that input variable values have certain relationships with the quality of the product produced, as measured by outer diameter. It is a classification/prediction problem.

Data was normalized to values between 0 and 1 for all inputs. Output was similarly normalized and then translated back to correct scale. A software simulation of the standard backpropagation algorithm as found in [4], except for the addition of a smoothing term during training, was used.

B. Sensitivity to Error Landscape Initiation Point

The first studies concentrated on how sensitive the networks were to where they begun on the error landscape. This sensitivity was measured in terms of number of epochs (training set cycles) to convergence within 10% for all training pairs and in terms of root mean square error (RMSE) for classification of a test set. The training set and the test set both totalled 44 samples, and were identical for all networks. A group of 10 networks of each configuration were trained and tested, all equivalent except for their error surface initiation point, which was completely arbitrary.

The first networks were trained to output the mean outer diameter as a continuously valued function while the second group used a binary representation, i.e. the output learned was a 1 for means beyond a 1 standard deviation limit about the diameter specification and a 0 for diameters within the 1 standard deviation limit. This second series of networks corresponded to a control chart formulation.

The continuous output group of networks had two hidden layers of 13 neurons each. Two network configurations were tried for the binary output; a one hidden layer with 13 neurons and a two hidden layer with 13 neurons in each layer. The sigmoid transfer function was altered between one ranging from 0 to 1 to one ranging from -0.5 to +0.5.

Table I shows the mean epochs to convergence, the coefficient of variation of epochs, the mean of the root mean square error of the trained networks, and the coefficient of variation of root mean square error for 10 networks of each kind. Coefficient of variation (C.V.) is the standard deviation of the sample divided by the mean and multiplied by 100 to form a percent. This measure of dispersion can be used regardless of the magnitude of the variables involved.

TABLE I
NETWORK VARIANCE IN RESPONSE TO ERROR SURFACE INITIATION

Output Type	Hidden Layers	Sigmoid	Mean Epochs	C.V. Epochs	Mean RMSE	C.V. RMSE
Continuous	2	0 to 1	2712	17.37%	.0035	6.91%
Binary	1	0 to 1	*	*	.3563	4.94%
Binary	2	0 to 1	2475	39.73%	.3503	3.41%
Binary	2	-.5 to .5	1775	31.32%	.3495	3.46%

* No networks converged.

Table I shows that while convergence is quite dependent on where the network initially begins, performance as measured by RMS error of the test set is very stable, though not identical. The binary outputs were more uniform in performance, as expected, since classification of inputs into one of two categories is a much simpler problem than returning a continuously valued output. Binary classification errors were fewer, but where errors occurred, they missed by a large margin, hence the greater RMS error. None of the networks had precisely the same performance on the test set, although they followed the same trends. Fig. 3 shows the outputs of 10 continuous networks over the test set, while Fig. 4 shows the same for 10 binary networks. For practical purposes, the networks performed the same.

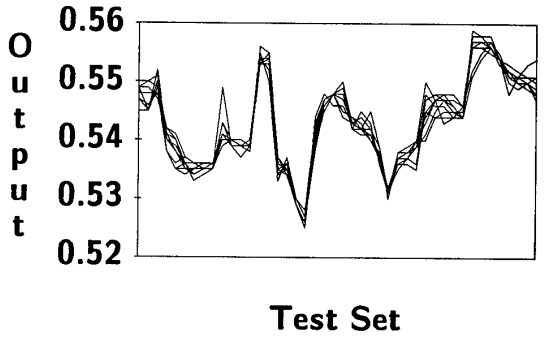


Fig. 3. Output of 10 continuous networks for the test set.

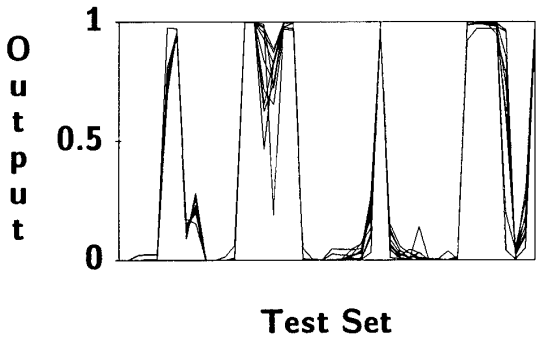


Fig. 4. Output of 10 binary networks for the test set.

The training rate affects how a network converges as discussed earlier. The two hidden layer network with continuous outputs and a sigmoid from 0 to 1 was tested with both a smaller and a larger training rate than used above, in conjunction with a smoothing rate of 0.9. A smaller training rate would be expected to lessen the sensitivity to error surface initiation in terms of performance since it more closely follows the surface. Conversely, convergence epochs would be expected to more sensitive with a small training rate since movement is slowed.

Table II shows the results for the network above compared with the two other training rates. The training rates of 0.5 and 2.0 resulted in greater epochs and greater variability in epochs. They did not result in significantly better performance or lesser variability in performance. The large coefficient of variance for convergence of the networks with n = 2 was primarily due to one network which finally converged after 19220 epochs. Note that both altered training rates resulted in one network which would not converge to within 10%, but still performed well.

TABLE II
NETWORK VARIANCE IN RESPONSE TO LEARNING RATE

Training Rate	Mean Epochs	C.V. Epochs	Mean RMSE	C.V. RMSE
n = 1	2712	17.37%	.0035	6.91%
n = 0.5	5728*	36.78%*	.0038	9.02%
n = 2	3974*	145.36%*	.0034	4.91%

* For 9 networks which converged.

The results indicate that training rate will affect epochs to convergence, but will have little impact on the network's ability to perform. These training rates are not extreme values, however, and are tempered by the smoothing factor.

C. *Weight Matrices*

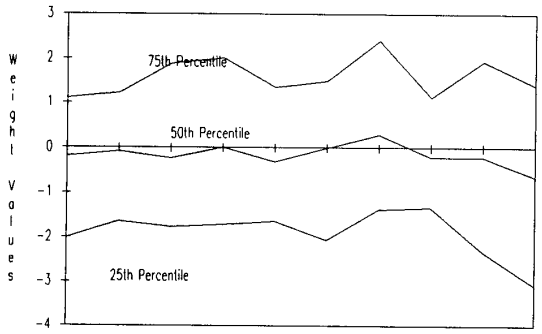
While performance was similar, the networks created during training were quite different. Weights from the first 10 continuous networks above were assembled and analyzed statistically. Weights which connected the input neurons to the first hidden layer neurons were one group, weights from the first hidden neuron layer to the second were another group, and weights from the second hidden layer to the output neuron were the third group. The first two groups totalled 182 weights each while the last had 14 weights (including the bias to each neuron). The statistical tests performed were a parametric one way analysis of variance (ANOVA) and a non-parametric one way ANOVA (the Kruskal-Wallis test). ANOVA is designed to test whether samples have different central tendencies. While it is not surprising that individual weights from different networks are different, it might be expected that collectively the weight matrices would be similar.

Table III shows the results of analysis for each of the three weight matrices from the 10 networks. Statistically, the two hidden layer weight matrices are different in both location and dispersion, while the output layer weights are not statistically different.

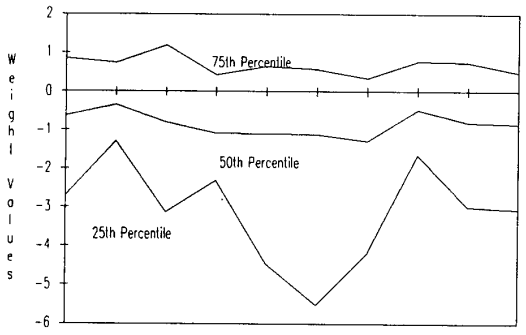
TABLE III
STATISTICAL TESTS FOR THREE WEIGHT MATRICES FROM 10 NETWORKS

Weight Matrix	F Test	P Value	Chi Square Equal Variance	P Value	Kruskal Wallis	P Value
Hid. 1	2.63	.0051	76.84	.0000	16.27	.0615
Hid. 2	3.78	.0001	145.29	.0000	30.93	.0003
Output	1.17	.3197	10.35	.3230	11.23	.2604

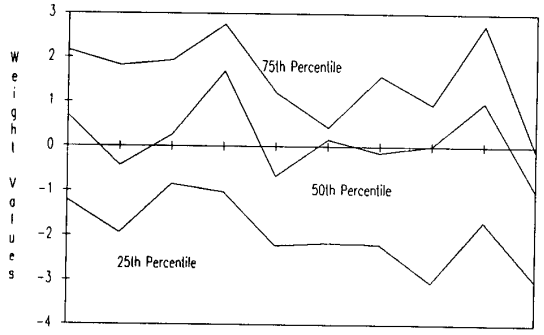
The weight matrices differences are shown pictorially in Fig. 5 where the 25th, 50th and 75th percentiles of each weight matrix are calculated for each of the 10 networks. These graphs show that there is quite a bit of variability in weight values among networks which differ only by their initiation point on the error landscape.



A. First hidden layer matrix.



B. Second hidden layer matrix.



C. Output layer matrix.

Fig. 5. Percentiles of weight matrices for 10 networks.

D. *Choosing Initial Weights*

As mentioned earlier, perhaps the best strategy that could be adopted before training a network is to intelligently select weights. In a practical sense, since the combinations of weight values which lead to similar network performance quickly reaches combinatorial explosion, even *a priori* knowledge of the problem would not be helpful. We decided to concentrate efforts for initial weight selection on the distribution of the weights rather than their actual values.

The continuous and binary outputs described in the preceding section had randomized beginning weights ranging from -1 to +1. In this section they all begin by alternating -0.5 and +0.5. Initial weights were increasingly randomized by adding a random amount to each weight with a Gaussian distribution and a standard deviation beginning at 0.1 and increasing in steps of 0.1 up to 0.9.

Table IV shows the results of these networks for various strategies. None of the networks with no noise in the weights of -0.5 or +0.5 converged. It can be seen by comparing Table I and Table IV that this scheme resulted in increased dispersion in both convergence and, especially, performance.

TABLE IV
NETWORK VARIANCE IN RESPONSE TO WEIGHT RANDOMIZATION

Output Type	Hidden Layers	Sigmoid	Mean Epochs	C.V. Epochs	Mean RMSE	C.V. RMSE
Continuous	2	0 to 1	4483*	29.79%	.0041	22.41%
Binary	2	0 to 1	4651*	60.68%	.3533	7.41%
Binary	2	-.5 to .5	2278*	38.31%	.3560	6.11%

* For the 9 networks which converged.

Fig. 6 shows the output of 10 binary networks. Comparing this to Fig. 4 shows that the output of these networks which differed in distribution of initial weights were more divergent than those from an identical distribution.

Fig. 7 shows how the RMSE of the output changes as the initial weights become more random for the continuously valued outputs, while Fig. 8 shows the same for the two binary valued output strategies. It is clear in the continuous network that some randomness in the initial weights improves final network performance. For the binary networks, randomness may not assist performance, but it does not hinder performance.

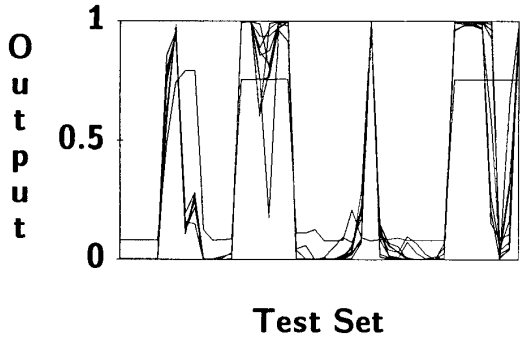


Fig. 6. Output of 10 binary networks with differing initial weight noise.

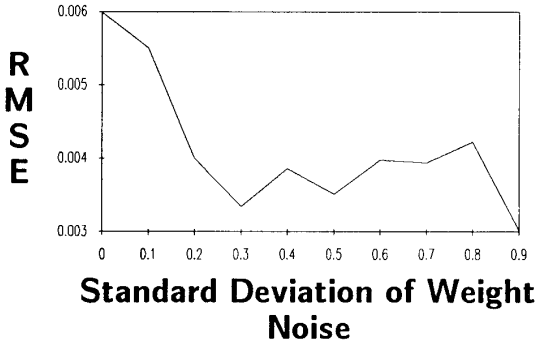


Fig. 7. RMSE of test set as initial weights are more random for continuous outputs.

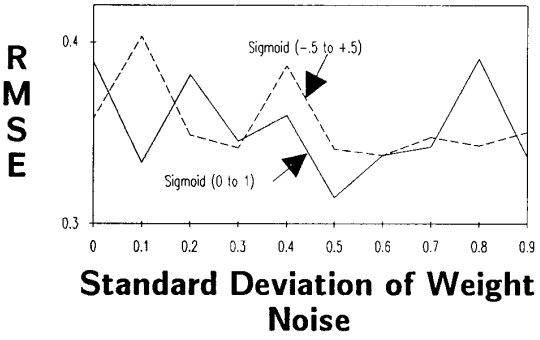


Fig. 8. RMSE of test set as initial weights are more random for two binary output formulations.

IV. CONCLUSIONS

Error landscape is an area of neural networks which is largely uncharted. For industrial implementation the important aspect is whether the beginning arbitrary weights significantly impact a typical network's ability to converge and to perform after training. For this pattern classification task, although weights derived are quite different and convergence epochs can also differ substantially, performance is relatively insensitive. However, performance is not equivalent among networks, empirically supporting the concept that most backpropagation error surfaces contain large, flat troughs near the bottom. Networks train to somewhere on the flat region, but not to its absolute gully. We are continuing exploration in the areas of sensitivity to network parameter changes and statistical characterization of weight matrices.

REFERENCES

[1] P. Baldi and K. Hornik, "Neural networks and principal component analysis: learning from examples without local minima," Neural Networks, vol. 2, pp. 53-58, 1989.

[2] R. Hecht-Nielsen, "Theory of the backpropagation neural network," Proceedings of the International Joint Conference on Neural Networks, pp. I-593-I-605, 1989.

[3] D. Ingman and Y. Merlis, "Local minima escape using thermodynamic properties of neural networks," Neural Networks, vol. 4, pp. 395-404, 1991.

[4] D. Rumelhart, J. McClelland, and the PDP Research Group, Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Cambridge, MA: MIT Press, 1986.

[5] P. Saratchandran, "Dynamic programming approach to optimal weight selection in multilayer neural networks," IEEE Transactions on Neural Networks, vol. 2, no. 4, pp. 465-467, 1991.

[6] W. Simon and J. Carter, "Back propagation learning equations from the minimization of recursive error," Proceedings of the IEEE International Conference on Systems Engineering, pp. 155-160, 1989.

[7] P. Wasserman, Neural Computing: Theory and Practice. New York: Van Nostrand Reinhold, 1989.

[8] M. Weir, "A method for self-determination of adaptive learning rates in back propagation," Neural Networks, vol. 4, pp. 371-379, 1991.

[9] N. Weymaere and J. Martens, "A fast and robust learning algorithm for feedforward neural networks," Neural Networks, vol. 4, pp. 361-369, 1991.

[10] R. Williams, On the use of backpropagation in associative reinforcement learning," Proceedings of the International Joint Conference on Neural Networks, pp. I-263-I-270, 1988.

[11] K. Yamada, H. Kami, J. Tsukumo, and T. Temma, "Handwritten numeral recognition by multi-layered neural network with improved learning algorithm," Proceedings of the International Joint Conference on Neural Networks, pp. II-259-II-266, 1989.