The Effect of Representation on Error Surface

Steven Phillips

Department of Computer Science,
The University of Queensland, QLD 4072 Australia. Email: stevep@cs.uq.oz.au

Abstract

Research on neural network learning within the supervised learning paradigm has focused on efficient search (or optimization) over the error surface. Less attention has been given to the effect representation has on the error surface. One interesting question to ask is, how does the choice of data points affect learning time for a neural network on linearly separable problems. This paper examines the issue of class representation in the light of its affect on error surface. Error surface plots visually suggest that an equal representation of points for each class decreases learning time. This hypothesis is supported by simulation results for a simple classification problem.

1 Introduction

In the supervised learning paradigm learning can be conceptualized as a search through weight space (guided by an error surface) for a region where the error is acceptably low. For backpropagation [10], the most commonly used network learning algorithm, search is performed by gradient descent on the error surface. Although backpropagation is a generally applicable algorithm it is considered too slow for large problems.

Improvements on backpropagation can be characterized as either more efficient traversal of an error surface, or, use of a better error surface. Examples of more efficient traversal algorithms are the quadratic surface approximators such as Quickprop [4] and conjugate gradient [3]. The error surface is determined in part by the network's activation function, error function and architecture. These three aspects have also been sources for improvement. Examples include, use of tanh [12] and gaussian [11] activation functions; cross-entropy [2] instead of sum of squares error function; and constructive/destructive algorithms such as Cascade Correlation [5].

The error surface is also determined by the representation of patterns (e.g., reachable targets [13]); the choice of training patterns (e.g., boundary points [8]); and the pattern selection strategy (e.g., *Repeat Until Bored* [9], *Selective Pattern Presentation* [6], and *Don't Care margins* [1]).

Issues regarding the benefits of representation are somewhat intuitive. Statements like 'ill-conditioned ravines', although true in some sense do not give the experimenter any better understanding of the effect representation has on learning. What this paper proposes is the judicious use of error surface plots to gain stronger intuitions (through visualization) as to the effect representation has on the error surface and consequently on learning. This paper investigates how the choice of data points representing the target classes affects error surface and therefore learning time.

In the second section, error surfaces were plotted for the two cases for a simple one-dimensional input space classification problem. From these plots an hypothesis was put forward which was tested by simulation in section 3. Section 4 discusses the implication of these results and conclusions are finally drawn.

2 Error surfaces

With all a networks complexity and flexibility comes the difficulty of understanding exactly what is does. McClelland and Rumelhart in [7] used two-dimensional error surface plots for a simple network to show the location of local minima in the error surface. Here three-dimensional error surfaces are plotted for a simple classification problem to demonstrate the effect the distribution of data points representing classes has on error surface.

Since error surface plots are limited to three-dimensions (two for the weights and one for the error), the classification problem was simplified to recognition of positive real numbers encoded in a one-dimensional input space by a network consisting of one unit with bias and weighted input using tanh as the activation function. The effect of class representation was examined on two extreme cases. The first case, called 1-N, containing one point from one class and N points from the other is representative of distributions heavily biased towards one of the classes. The second case, called N-N, represents a more even distribution of data

points. For the 1-N case the following data set was used

$f(-0.1) \rightarrow 0, f(0.1) \rightarrow 1, f(1.0) \rightarrow 1, f(2.0) \rightarrow 1.$

For the N-N case,

$f(-2.0) \rightarrow 0, f(-1.0) \rightarrow 0, f(-0.1) \rightarrow 0,$

$f(-0.1) \rightarrow 0, f(0.1) \rightarrow 1, f(1.0) \rightarrow 1.$

The error surfaces resulting from the two cases is shown in figures 1 and 2 respectively.

The error surface for the 1-N case consisted of four ridges and a ravine. The solution space (region of weight space where all inputs are correctly classified) lies in the ravine. What is interesting about the ridges is their faces in general point away from the solution space. A search based on local gradient information is likely to move away from the solution space to the flat region before heading back in the direction of the solution. A gradient descent technique which moves through the weight space in proportion to the gradient (like backpropagation) will take considerable time to find its way back to the solution space. Comparing this to the second case the effect of even distribution between classes is evident. The previous flat region now includes additional ridges pointing in the direction of the solution space.

A visualization of the two error surfaces was suggestive of the following hypothesis: *learning from a more even between-class distribution of training points is faster than a biased between-class distribution.* A simple simulation was conducted to support this hypothesis.

3 Simulation

The error surface plots although confined to a one-dimensional input space were qualitative visual descriptions of the effect between-class distribution of training points has on the shape of the error surface. To gain support for the hypothesis arising from the plots and its applicability in general to classification over N-dimensional input domains, simulations were run on a network trained to recognize the first 1024 natural numbers encoded as 10 bit binary numbers. The input space was 10-dimensional with three allowable values (-1, 0, 1) along each dimension. Given an input pattern the network was required to determine whether it belonged to the subset of natural numbers.

The network was trained on two different data sets. The first set contained the smallest number of points necessary to learn correct classification[1] (1-N data set). This is an example of an uneven between-class distribution of points in N-dimensional space. The second set contained an almost equal number of boundary points from both classes and is termed the N-N data set [2]. This is an example of a relatively even between-class distribution of points.

The network was initialized with random weights between -1 and 1 and trained with a learning rate of 0.1 and a momentum of 0.0 and 0.5. Training stopped when the network output was on the right side of 0.5. Weights were updated at the end of each epoch (i.e., after each pattern was presented). Table 1 shows the mean and standard deviation over 20 trials of the number of pattern presentations before successful recognition of all patterns.

Table 1: Mean(StDev) training time over 20 trials

Data set	Momentum=0.0	Momentum=0.5
1-N	9639(1913)	4513(908)
N-N	444(38)	248(39)

4 Discussion

The error surface plots suggested learning would be faster when both classes where equally represented. Mean training time for the two data sets with and without momentum supported this prediction. Simply using a momentum term although improving the absolute learning time for each data set did not change the relationship between the mean learning time for the two data sets. With momentum at 0.0 and 0.5, mean learning times were significantly faster, $t(38) = 20.9$, $p < .001$ and $t(38) = 19.7$, $p < .001$, respectively, with the N-N data set. This suggests that although momentum decreases the time spent moving across flat regions it does not avoid them.

A network was trained twice, once on each of the two data sets starting from the same set of weights. The average sum of squares error per pattern versus number of training patterns presented was plotted for each training run (figure 3). In both cases most of the average error was reduced quickly, however in the N-N case error continued to steadily decrease whereas in the 1-N case further reduction in error was very slow confirming the presence of a flat region. The error surface plots suggest that the search moved into the flat region because of the direction of the ridges. These ridges are a consequence of the training patterns. An uneven distribution of training patterns results in ridges in the error surface which when added together point away from the solution space toward a flat region of the error surface.

[1] The smallest set contains the 10 boundary points encoded with a one and nine zeros as representatives of the class of naturals plus the zero vector representing a boundary point from the other class.

[2] There are actually N+1 points in one class as the zero point is also included

The implication of this result is that classes should have roughly equal representation in terms of the number of points from the input space. The result is not specific to the optimization technique. It simply applies to the error surface being searched. Any technique which uses local gradient information should show relative improvement when using evenly distributed data.

5 Conclusion

Error surface plots although limited in the number of dimensions that can be shown at once, if used judiciously, visually provide strong intuitions as to network learning behaviour. In this paper, error surface plots were used to gain an understanding of the effect choice of data points representing the target classes has on error surface and consequently learning time. The error surface plots suggested learning time would improve if the target classes were equally represented. This hypothesis was supported by a simulation experiment.

Acknowledgement

The author was supported by a University of Queensland Postgraduate Research Award.

References

[1] Paul Bakker. Dont care margins help backpropagation learn exceptions. In *AI'92*, November 1992. To be presented.

[2] Eric B. Baum and Frank Wilczek. Supervised learning of probability distributions by neural networks. In Dana Z. Anderson, editor, *Neural Information Processing Systems*, pages 52–61, 335 E. 45th St., New York, 1987. American Institute of Physics, AIP.

[3] Sue Becker and Yann LeCun. The feasibility of applying numerical optimization techniques to back-propagation. In *1988 Connectionist Summer School*, 1988.

[4] Scott Fahlman. An empirical study of back-propagation networks. Technical Report CMU-CS-88-162, Carnegie-Mellon University, 1988.

[5] Scott E Fahlman and Christian Lebiere. The cascade-correlation learning algorithm. Technical Report CMU-CS-90-100, Carnegie Mellon University, 1990.

[6] Kazuhiro Kohara. Selective presentation learning for pattern recognition by back-propagation neural networks. In Philip Leong and Marwan Jabris, editors, *Proceedings of the Third Australian Conference on Neural Networks*, pages 190–193, Canberra, February 1992. Sydney University Electical Engineering.

[7] James L McClelland and David E Rumelhart. *Training Hidden Units: The Generalized Delta Rule*, volume 3 of *Explorations in Parallel Distributed Processing*, chapter 5, pages 121–159. MIT Press, 1988.

[8] Kishan G Mehrotra, Chiluluri K Mohan, and Sanjay Ranka. Bounds on the number of samples needed for neural learning. *IEEE Transactions on Neural Networks*, 2(6):548–558, 1991.

[9] Paul W Munro. Repeat until bored: A pattern selection strategy. In John E Moody, Steven J Hanson, and Richard P Lippmann, editors, *Advances in Neural Information Processing Sytems 4*, pages 1001–1008, San Mateo, California, 1992. Morgan Kaufmann.

[10] D E Rumelhart, G E Hinton, and R J Williams. *Learning Internal Representations by Error Propagation*, volume 2 of *Explorations in Parallel Distributed Processing*, chapter 8, pages 319–362. MIT Press, 1986.

[11] Robert M Sanner and Jean-Jacques E Slotine. Direct adaptive control using guassian networks. Technical Report NSL-910303, MIT, March 1991.

[12] W Scott Stornetta and B A Huberman. An improved three-layer back-propagation algorithm. In Maureen Caudill and Charles Butler, editors, *IEEE First Internation Conference on Neural Networks*, pages 637–645, San Diego, CA, 1987. IEEE.

[13] David S Touretzky and Dean A Pomerleau. Neural networks for real-world problems. School of Computer Science, CMU, Pittsburgh, PA 15213, 1991. Tutorial T.4 of the 12th International Conference on Artificial Intelligence.

Appendix Figures

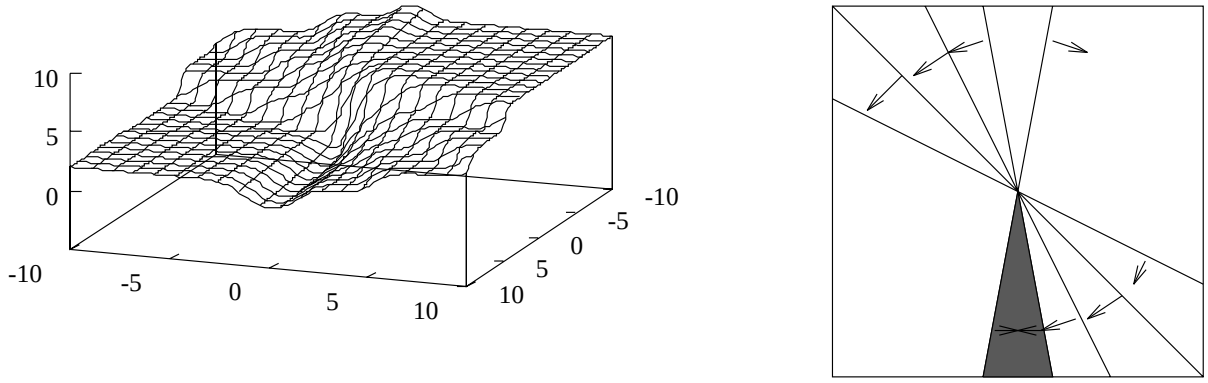

Figure 1 3-D plot of error surface for the 1-N data set. Plan view of suface showing the solution space (shaded region), direction of slopes amd the predominant direction around the center of weight space.

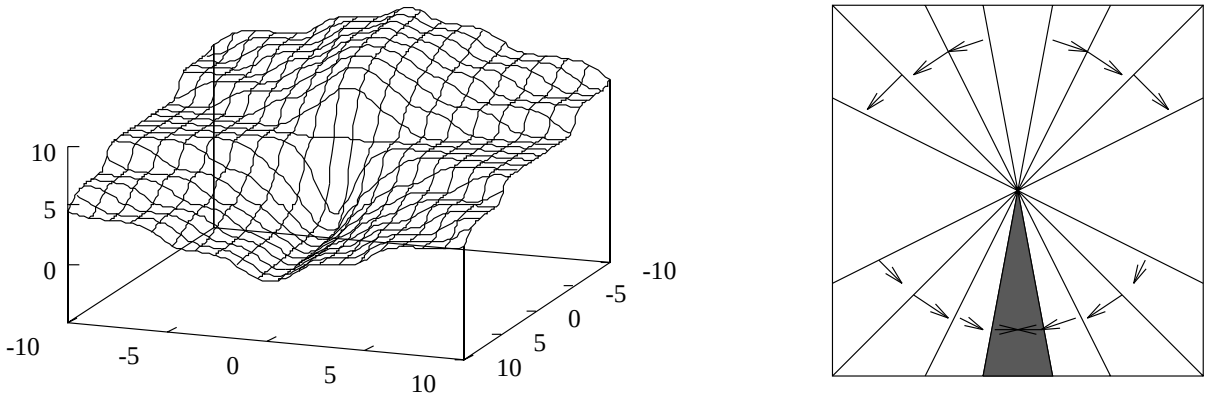

Figure 2 3-D plot of error surface for the N-N data set. Plan view of suface showing the solution space (shaded region) , direction of slopes amd the predominant direction around the center of weight space.

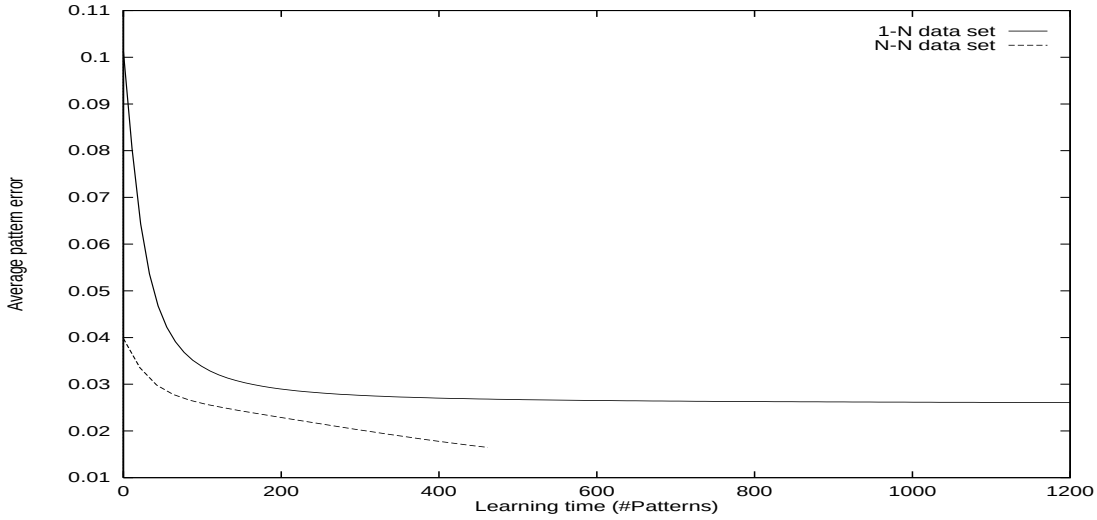

Figure 3 Plot of average pattern versus number of training patterns.