Complex Systems **2** (1988) 625–640

Accelerated Learning in Layered Neural Networks

Sara A. Solla
AT&T Bell Laboratories, Holmdel, NJ 07733, USA

Esther Levin
Michael Fleisher
Technion Israel Institute of Technology, Haifa 32000, Israel

Abstract. Learning in layered neural networks is posed as the minimization of an error function defined over the training set. A probabilistic interpretation of the target activities suggests the use of relative entropy as an error measure. We investigate the merits of using this error function over the traditional quadratic function for gradient descent learning. Comparative numerical simulations for the *contiguity problem* show marked reductions in learning times. This improvement is explained in terms of the characteristic steepness of the landscape defined by the error function in configuration space.

1. Introduction

Categorization tasks for which layered neural networks can be trained from examples include the case of probabilistic categorization. Such implementation exploits the analog character of the activity of the output units: the activity $\mathcal{O}_j^\alpha$ of the output neuron j under presentation of input pattern α is proportional to the probability that input pattern α possesses attribute j. The interpretation of target activities as probability distributions leads to measuring network performance in processing input α by the relative entropy of the target to the output probability distributions [1–3].

There is no *a priori* way of deciding whether such a logarithmic error function can improve the performance of a gradient descent learning algorithm, as compared to the standard quadratic error function. To investigate this question we performed a comparative numerical study on the contiguity problem, a classification task used here as a test case for the efficiency of learning algorithms.

We find that measuring error by the relative entropy leads to notable reductions in the time needed to find good solutions to the learning problem. This improvement is explained in terms of the characteristic steepness of the surface defined by the error function in configuration space.

The paper is organized as follows: The learning problem is discussed in section 2. Comparative numerical results for the contiguity problem are presented in section 3, and section 4 contains a summary and discussion of our results.

2. The Learning Problem

Consider a layered feed-forward neural network with deterministic parallel dynamics, as shown in figure 1.

The network consists of $L+1$ layers providing L levels of processing. The first layer with $\ell = 0$ is the input field and contains N_0 units. Subsequent layers are labeled $1 \leq \ell \leq L$; the ℓ^{th} layer contains N_ℓ units. The $(\ell+1)^{\text{th}}$ level of processing corresponds to the state of the units in layer ℓ determining that of the units in layer $(\ell+1)$ according to the following deterministic and parallel dynamical rule:

$$\begin{aligned} U_i^{(\ell+1)} &= \sum_{j=1}^{N_\ell} W_{ij}^{(\ell+1)} V_j^{(\ell)} + W_i^{(\ell+1)}, \\ V_i^{(\ell+1)} &= g(U_i^{(\ell+1)}). \end{aligned} \tag{2.1}$$

The input $U_i^{(\ell+1)}$ to unit i in layer $(\ell+1)$ is a linear combination of the states of the units in the preceding layer. This input determines the state $V_i^{(\ell+1)}$ via a nonlinear, monotonically increasing, and bounded transfer function $V = g(U)$.

The first layer receives input from the external world: a pattern is presented to the network by fixing the values of the variables $V_i^{(0)}$, $1 \leq i \leq N_0$. Subsequent layer states are determined consecutively according to equation (1). The state of the last layer $\ell = L$ is interpreted as the output of the network. The network thus provides a mapping from input space $\vec{V}^{(0)}$ into output space $\vec{V}^{(L)}$.

The network architecture is specified by the number $\{N_\ell\}, 1 \leq \ell \leq L$ of units per layer, and its configuration by the couplings $\{W_{ij}^{(\ell)}\}$ and biases $\{W_i^{(\ell)}\}$ for $1 \leq \ell \leq L$. Every point $\{\vec{W}\}$ in configuration space represents a specific network design which results in a specific input-output mapping. The design problem is that of finding points $\{\vec{W}\}$ corresponding to a specific mapping $\vec{V}^{(L)} = f(\vec{V}^{(0)})$.

Consider a labeling of the input vectors by an index α. The input vectors $\vec{I}^\alpha = \vec{V}^{(0)}$ are usually binary, and $1 \leq \alpha \leq 2^{N_0}$. The output vector $\vec{O}^\alpha = \vec{V}^{(L)}$ corresponding to input $\vec{I}^\alpha$ is determined by $\vec{O}^\alpha = f(\vec{I}^\alpha)$ for all α.

The network can be used for classification, categorization, or diagnosis by interpreting the activity O_j^α of the j^{th} output unit under presentation of the α^{th} input pattern as the probability that the input belongs to category j or possesses the j^{th} attribute. This interpretation requires activity values O_j^α restricted to the [0,1] interval. If the range [a,b] of the transfer function $V = g(U)$ is different from [0,1], the outputs are linearly rescaled to the

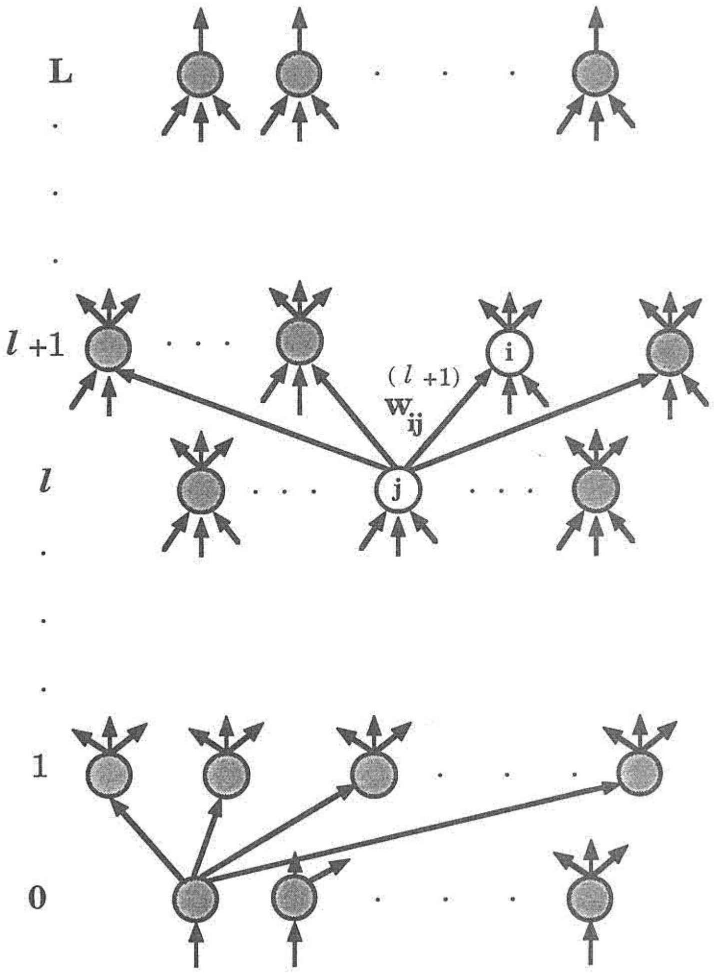


Figure 1: A layered feed-forward network with L levels of processing.

desired interval. Note that the class of Boolean mappings corresponds to output units restricted to binary values $\mathcal{O}_j^\alpha = 0, 1$, and is a subclass of the general class of mappings that can be implemented by such networks.

A quantitative formulation of the design problem of finding an appropriate network configuration $\{\vec{W}\}$ to implement a desired mapping requires a measure of quality: to which extent does the mapping realized by the network coincide with the desired mapping? The dissimilarity between these two mappings can be measured as follows. Given a subset of input patterns $\vec{\mathcal{I}}^\alpha, 1 \leq \alpha \leq m$ for which the desired outputs $\vec{\mathcal{T}}^\alpha, 1 \leq \alpha \leq m$ are known, and given the outputs $\vec{\mathcal{O}}^\alpha = f(\vec{\mathcal{I}}^\alpha)$ produced by the network, the distance

$$d^\alpha = d(\vec{\mathcal{O}}^\alpha, \vec{\mathcal{T}}^\alpha) \tag{2.2}$$

between the target $\vec{\mathcal{T}}^\alpha$ and the output $\vec{\mathcal{O}}^\alpha$ measures the error made by the network when processing input $\vec{\mathcal{I}}^\alpha$. Then the error function

$$E \equiv \sum_{\alpha=1}^{m} d^\alpha = \sum_{\alpha=1}^{m} d(\vec{\mathcal{O}}^\alpha, \vec{\mathcal{T}}^\alpha) \tag{2.3}$$

measures the dissimilarity between the mappings on the restricted domain of input patterns $\{\vec{\mathcal{I}}^\alpha\}, 1 \leq \alpha \leq m$. For a fixed set of pairs $\{\vec{\mathcal{I}}^\alpha, \vec{\mathcal{T}}^\alpha\}, 1 \leq \alpha \leq m$, the function E is an implicit function of the couplings $\{\vec{W}\}$ through the outputs $\{\vec{\mathcal{O}}^\alpha\}$.

There are many possible choices for $d(\vec{\mathcal{O}}^\alpha, \vec{\mathcal{T}}^\alpha)$. The only required property is[1]

$$d(\vec{\mathcal{O}}^\alpha, \vec{\mathcal{T}}^\alpha) \geq 0 \tag{2.4}$$

and

$$d(\vec{\mathcal{O}}^\alpha, \vec{\mathcal{T}}^\alpha) = 0 \qquad \text{iff} \qquad \vec{\mathcal{O}}^\alpha = \vec{\mathcal{T}}^\alpha, \tag{2.5}$$

which guarantees

$$E \geq 0 \tag{2.6}$$

and

$$E = 0 \qquad \text{iff} \qquad \vec{\mathcal{O}}^\alpha = \vec{\mathcal{T}}^\alpha \qquad\qquad \text{for all } \alpha, \ 1 \leq \alpha \leq m. \tag{2.7}$$

The design problem thus becomes that of finding global minima of $E(\vec{W})$: the configuration space $\{\vec{W}\}$ has to be searched for points that satisfy

$$E(\vec{W}) = 0. \tag{2.8}$$

This design method of training by example extracts information about the desired mapping from the training set $\{\vec{\mathcal{I}}^\alpha, \vec{\mathcal{T}}^\alpha\}, 1 \leq \alpha \leq m$.

A standard method to solve the minimization problem of equation (2.8) is gradient descent. Configuration space $\{\vec{W}\}$ is searched by iterating

[1]The term 'distance' is not used here in its rigorous mathematical sense, in that symmetry and triangular inequality are not required.

$$\Delta\vec{W} = -\eta\ \nabla_{\vec{W}} E(\vec{W}) \tag{2.9}$$

from an arbitrary starting point $\vec{W}_o$. This algorithm requires differentiability of the transfer function $V = g(U)$, and of the distance $d(\vec{\mathcal{O}}^\alpha, \vec{\mathcal{T}}^\alpha)$ with respect to the output variables $\vec{\mathcal{O}}^\alpha$.

Many choices of $d(\vec{\mathcal{O}}^\alpha, \vec{\mathcal{T}}^\alpha)$ satisfy the listed requirements. The sum of squared errors

$$d(\vec{\mathcal{O}}, \vec{\mathcal{T}}) = \frac{1}{2} \parallel \vec{\mathcal{O}} - \vec{\mathcal{T}} \parallel^2 \tag{2.10}$$

leads to a quadratic error function

$$E_Q = \frac{1}{2} \sum_{\alpha=1}^{m} \sum_{j=1}^{N_L} (\mathcal{O}_j^\alpha - \mathcal{T}_j^\alpha)^2, \tag{2.11}$$

and to the generalized δ-rule [4] when used for gradient descent. Although in most existing applications of the δ-rule [4,5] the target activities of the output units are binary, $\mathcal{T}_j^\alpha = 0, 1$, the algorithm can also be used for probabilistic targets $0 \leq \mathcal{T}_j^\alpha \leq 1$. But given a probabilistic interpretation of both outputs $\vec{\mathcal{O}}^\alpha$ and targets $\vec{\mathcal{T}}^\alpha$, it is natural to choose for $d(\vec{\mathcal{O}}^\alpha, \vec{\mathcal{T}}^\alpha)$ one of the standard measures of distance between probability distributions. Here we investigate the properties of a logarithmic measure, the entropy of $\vec{\mathcal{T}}$ with respect to $\vec{\mathcal{O}}$ [6].[2]

A binary random variable x_j associated with the jth output unit describes the presence (x_j=True) or absence (x_j=False) of the jth attribute. For a given input pattern $\vec{I}^\alpha$, the activity $\vec{\mathcal{O}}^\alpha$ reflects the conditional probabilities

$$P\{x_j = \mathrm{T}|\alpha\} = \mathcal{O}_j^\alpha \tag{2.12}$$

and

$$P\{x_j = \mathrm{F}|\alpha\} = \overline{\mathcal{O}_j^\alpha} = 1 - \mathcal{O}_j^\alpha. \tag{2.13}$$

$\mathcal{T}_j^\alpha$ and $\overline{\mathcal{T}_j^\alpha}$=1-$\mathcal{T}_j^\alpha$ are the target values for these conditional probabilities. The relative entropy of target with respect to output [6] is given by

$$d_j^\alpha = \mathcal{T}_j^\alpha \ \ln \frac{\mathcal{T}_j^\alpha}{\mathcal{O}_j^\alpha} + (1 - \mathcal{T}_j^\alpha) \ \ln \frac{(1 - \mathcal{T}_j^\alpha)}{(1 - \mathcal{O}_j^\alpha)} \tag{2.14}$$

for each output unit, $1 \leq j \leq N_L$. As a function of the output variable $\mathcal{O}_j^\alpha$, the function d_j^α is differentiable, convex, and positive. It reaches its global minimum of $d_j^\alpha = 0$ at $\mathcal{O}_j^\alpha = \mathcal{T}_j^\alpha$. Then

$$d^\alpha = d(\vec{\mathcal{O}}^\alpha, \vec{\mathcal{T}}^\alpha) = \sum_{j=1}^{N_L} d_j^\alpha \tag{2.15}$$

[2]Other choices for measuring distances between probability distributions are available, such as the Bhattacharyya distance [7].

leads to a logarithmic error function

$$E_L = \sum_{\alpha=1}^{m} \sum_{j=1}^{N_L} \left\{ \mathcal{T}_j^\alpha \, \ln \frac{\mathcal{T}_j^\alpha}{\mathcal{O}_j^\alpha} + (1 - \mathcal{T}_j^\alpha) \, \ln \frac{(1 - \mathcal{T}_j^\alpha)}{(1 - \mathcal{O}_j^\alpha)} \right\}. \tag{2.16}$$

A similar function has been proposed by Baum and Wilczek [1] and Hinton [3] as a tool for maximizing the likelihood of generating the target conditional probabilities. Their learning method requires maximizing a function $\tilde{E}$ given by

$$\tilde{E}(\vec{W}) = -\left(E_L(\vec{W}) + E_0\right), \tag{2.17}$$

where $E_L(\vec{W})$ is the error function defined by equation (2.16), and the constant

$$E_0 = \sum_{\alpha=1}^{m} \sum_{j=1}^{N_L} \left\{ \mathcal{T}_j^\alpha \, \ln \frac{1}{\mathcal{T}_j^\alpha} + (1 - \mathcal{T}_j^\alpha) \, \ln \frac{1}{(1 - \mathcal{T}_j^\alpha)} \right\} \tag{2.18}$$

is the entropy of the target probability distribution for the full training set. Extrema of the function $\tilde{E}(\vec{W})$ do not correspond to $\tilde{E}(\vec{W}) = 0$, but to $\tilde{E}(\vec{W}) = -E_0$, a quantity that depends on the targets in the training set. It is preferable to use the normalization of $E_L(\vec{W})$ in equation (2.16), to eliminate this dependence on the target values.

The error function $E_L(\vec{W})$ of equation (2.16) has been used by Hopfield [2] to compare layered network learning with Boltzmann machine learning.

There is no a priori way of selecting between the quadratic error function E_Q and logarithmic measures of distance such as the relative entropy E_L. The efficiency of gradient descent to solve the optimization problem $E(\vec{W}) = 0$ is controlled by the properties of the error surface $E(\vec{W})$, and depends on the choice of distance measure and nonlinear transfer function.

The generalized δ-rule of Rumelhart et al. [4] results from applying gradient descent (equation (2.9)) to the quadratic error function $E_Q(\vec{W})$. When applied to the relative entropy $E_L(\vec{W})$, gradient descent leads to a similar algorithm, summarized as follows:

$$\Delta W_{ij}^{(\ell+1)} = \eta \; \delta_i^{(\ell+1)} \; V_j^{(\ell)} \tag{2.19}$$

and

$$\Delta W_i^{(\ell+1)} = \eta \; \delta_i^{(\ell+1)}, \tag{2.20}$$

where the error $\delta_i^{(\ell)}$ is defined as

$$\delta_i^{(\ell)} \equiv -\frac{\partial E}{\partial U_i^{(\ell)}} = -g'(U_i^{(\ell)}) \, \frac{\partial E}{\partial V_i^{(\ell)}}. \tag{2.21}$$

These equations are identical to those in reference [4], and are valid for any differentiable error function. Following reference [4], the weights are updated according to equations (2.19) and (2.20) after presentation of *each* training pattern $\vec{I}^{\alpha}$.

The calculation of $\delta_i^{(L)}$ for units in the top level depends explicitly on the form of $E(\vec{W})$, since $V_i^{(L)} = \mathcal{O}_i$. For $E_L(\vec{W})$,

$$\delta_i^{(L)} = g'(U_i^{(L)}) \, \frac{\mathcal{T}_i^{\alpha} - \mathcal{O}_i^{\alpha}}{\mathcal{O}_i^{\alpha}(1 - \mathcal{O}_i^{\alpha})}. \tag{2.22}$$

For hidden units in levels $1 \le \ell \le L-1$,

$$\delta_i^{(\ell)} = g'(U_i^{(\ell)}) \sum_{k=1}^{N_{\ell+1}} \delta_k^{(\ell+1)} W_{ki}^{(\ell+1)}, \tag{2.23}$$

the standard rule for error back-propagation [4]. A simplification results from choosing the logistic transfer function,

$$V = g(U) = \frac{1}{2}(1 + \tanh \frac{1}{2} U) = (1 + e^{-U})^{-1}, \tag{2.24}$$

for which there is a cancellation between the derivative $g'(U)$ and the denominator of equation (2.22), leading to

$$\delta_i^{(L)} = \mathcal{T}_i^{\alpha} - \mathcal{O}_i^{\alpha} \tag{2.25}$$

for the error at the output level.

This algorithm is quite similar to the generalized δ-rule, and there is no a priori way of evaluating their relative efficiency for learning. We investigate this question by applying them both to the contiguity problem [8,9]. This classification task has been used as a test problem to investigate learning abilities of layered neural networks [8–11].

3. A test case

The contiguity problem is a classification of binary input patterns $\vec{I} = (I_1, \ldots, I_N)$, $I_i = 0, 1$ for all $1 \le i \le N$, into classes according to the number k of blocks of +1's in the pattern [8,9]. For example, for $N = 10$, $\vec{I} = (0110011100)$ corresponds to $k = 2$, while $\vec{I} = (0101101111)$ corresponds to $k = 3$. This classification leads to $(1 + k_{\max})$ categories corresponding to $0 \le k \le k_{\max}$, with $k_{\max} = \mid \frac{N}{2} \mid$, where $\mid x \mid$ refers to the upper integer part [3] of x. There are

$$\mathcal{N}(k) = \binom{N+1}{2k} \tag{3.1}$$

[3]For any real number x in the interval $(n-1) < x \le n$, $\mid x \mid = n$ is the upper integer part of x.

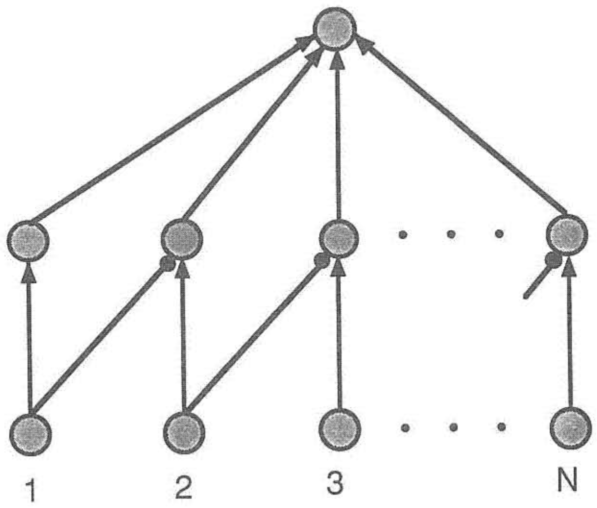


Figure 2: A solution to the contiguity problem with $L = 2$. All couplings $\{W_{ij}\}$ have absolute value of unity. Excitatory ($W_{ij} = +1$) and inhibitory ($W_{ij} = -1$) couplings are indicated by $\rightarrow$ and $-\!\bullet$, respectively. Intermediate units are biased by $W_i^{(1)} = -0.5$; the output unit is biased by $W_i^{(2)} = -(k_0 + 0.5)$.

input patterns in the k^{th} category. A simpler classification task investigated here is the dichotomy into two classes corresponding to $k \leq k_0$ and $k > k_0$. This problem can be solved [8,9] by an $L = 2$ layered network with $N_0 = N$, $N_1 = N$, and $N_2 = 1$. The architecture is shown in figure 2. The first level of processing detects subsequent 01 pairs corresponding to the left edge of a block, and the second level of processing counts the number of such edges.

Knowing that such solution exists, we choose for our learning experiments the network architecture shown in figure 3, with $N_0 = N_1 = N$ and $N_2 = 1$. The network is not fully connected: each unit in the $\ell = 1$ layer receives input from only the p subsequent units just below it in the $\ell = 0$ layer. The parameter p can be interpreted as the width of a *receptive field*.

The results reported here are for $N = 10$, $k_0 = 2$, and $2 \leq p \leq 6$. The total domain of $2^N = 1024$ input patterns was restricted to the union of $\mathcal{N}(2) = 330$ and $\mathcal{N}(3) = 462$ corresponding to $k = 2$ and $k = 3$ respectively. Out of these $\mathcal{N}(2) + \mathcal{N}(3) = 792$ input patterns, a training set of $m = 100$ patterns was randomly selected, with $m(2) = m(3) = 50$ examples of each category.

The starting point $\vec{W}_o$ for the gradient descent algorithm was chosen at random from a normal distribution. The step size η for the downhill search was kept constant during each run.

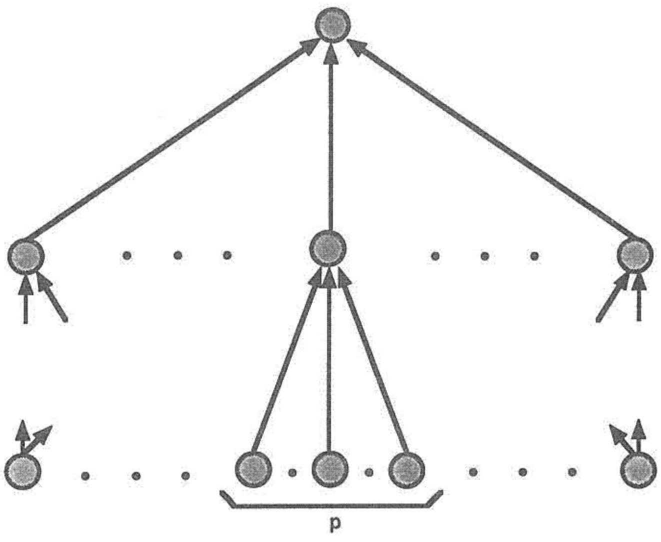


Figure 3: Network architecture for learning contiguity, with $L = 2$, $N_0 = N_1 = N$, $N_2 = 1$, and a receptive field of width p.

Numerical simulations were performed in each case from the *same* starting point $\vec{W}_o$ for both $E_Q(\vec{W})$ and $E_L(\vec{W})$. After each learning iteration consisting of a full presentation of the training set ($\Delta t = 1$), the network was checked for learning and generalization abilities by separately counting the fraction of correct classifications for patterns within ($\%L$) and not included ($\%G$) in the training set, respectively. The classification of the α^{th} input pattern is considered correct if $|\mathcal{O}^\alpha - \mathcal{T}^\alpha| \leq \Delta$, with $\Delta = 0.1$.

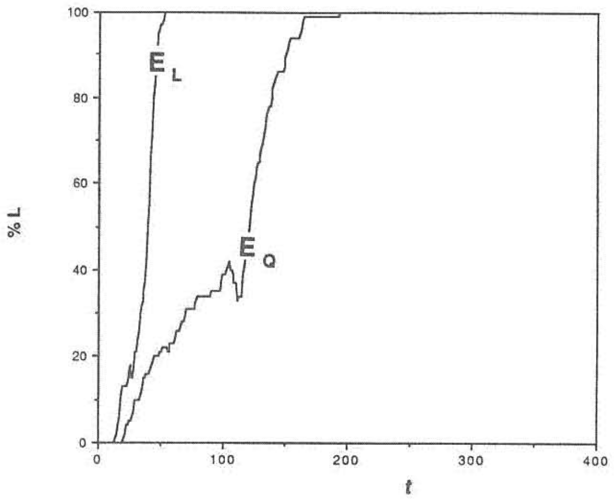
Both the learning $\%L$ and generalization $\%G$ abilities of the network are monitored as a function of time t. The learning process is terminated after τ presentations of the training set, either because $\%G = 100$ has been achieved, or because the network performance on the training set, as measured by

$$\varepsilon = \sum_{\alpha=1}^{m} (\mathcal{O}^\alpha - \mathcal{T}^\alpha)^2, \tag{3.2}$$

satisfies the stopping criterion $\varepsilon \leq 0.01$.

Comparative results between the two error functions are illustrated for $p = 2$ in figure 4. In both cases the networks produced by learning all patterns in the training set ($\%L = 100$) exhibited perfect generalization ability ($\%G = 100$). Note the significant reduction in learning time for the relative entropy with respect to the traditional quadratic error function. This accelerated learning was found in all cases.

Results summarized in table 1 correspond to several simulations for each error function, all of them successful in learning ($\%L = 100$). Results for



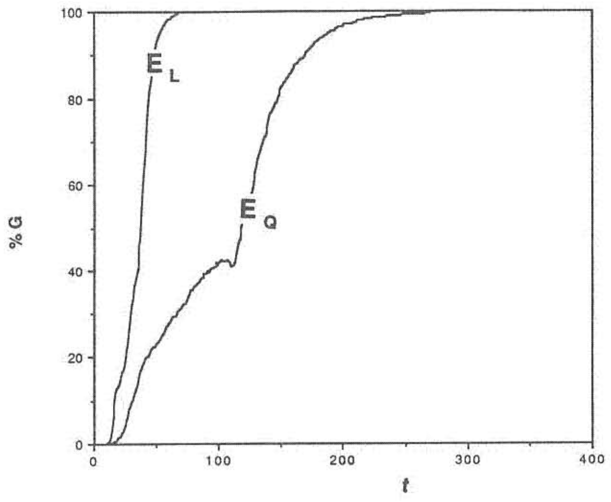


Figure 4: Learning and generalization ability as a function of time t for $p = 2$. Comparative results are shown for $E_L(\vec{W})$ and $E_Q(\vec{W})$.

p	E_Q		E_L	
	$\% G$	τ	$\% G$	τ
2	100	339	100	84
3	95	1995	91	593
4	90	2077	81	269
5	73	2161	74	225
6	68	2147	68	189

Table 1: Comparative results for learning contiguity with E_L and E_Q, averaged over successful runs ($\% L = 100$). Learning time τ is measured by the number of presentations of the full training set needed to achieve $\varepsilon \leq 0.01$ or $\% G = 100$.

E_Q			E_L		
$\% G$	$\% L$	τ	$\% G$	$\% L$	τ
83	95	1447	91	100	1445

Table 2: Comparative results for learning contiguity with E_L and E_Q, averaged over all runs for $p = 2$.

each value of p are averages over 10 runs with different starting points $\vec{W}_o$. The monotonic decrease in generalization ability with increasing p previously observed for the quadratic error function [9] is also found here for the relative entropy. Using this logarithmic error function for learning does not improve the generalization ability of the resulting network. Its advantage resides in systematic reductions in learning time τ, as large as an order of magnitude for large p.

Learning contiguity with the network architecture of figure 3 is always successful for $p \geq 3$: for both E_L and E_Q the learning process leads to configurations that satisfy $E(\vec{W}) = 0$ for all tried starting points. For $p = 2$, the gradient descent algorithm often gets trapped in local minima with $E(\vec{W}) > 0$, and $\% L = 100$ cannot be achieved. As shown by the results in table 2, the gradient descent algorithm is less likely to get trapped in local minima when applied to $E_L(\vec{W})$.

We conjecture that a larger density of local minima that are not global minima in $E_Q(\vec{W})$ is responsible for the more frequent failure in learning for $p = 2$. This conjecture is supported by the fact that learning contiguity with E_L required a smaller step size η than for E_Q. A value of $\eta = 0.25$ used for all runs corresponding to $E_Q(\vec{W})$ led to oscillations for $E_L(\vec{W})$. That a smaller value of $\eta = 0.125$ was required in this case indicates a smoother error function, with less shallow local minima and narrower global minima, as schematically illustrated in figure 5.

A decrease in the density of local minima for the logarithmic error function $E_L(\vec{W})$ is due in part to the cancellation between $g'(U^{(L)})$ and $\mathcal{O}(1 - \mathcal{O})$ in equation (2.22) for $\delta^{(L)}$. Such cancellation requires the use of the logistic

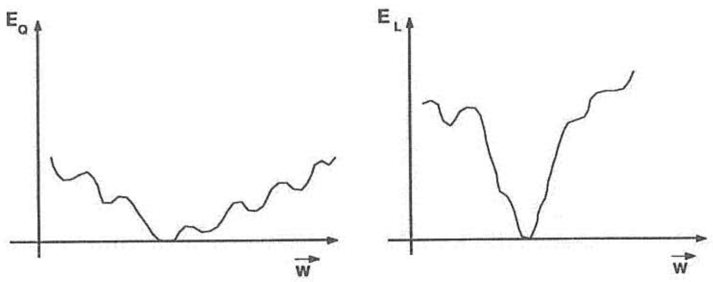


Figure 5: Schematic representation of the landscape defined by the error functions E_L and E_Q in configuration space $\{\vec{W}\}$.

transfer function of equation (2.24), and eliminates the spurious minima that otherwise arise for $\mathcal{O} = 1$ when $\mathcal{T} = 0$, and for $\mathcal{O} = 0$ when $\mathcal{T} = 1$.

4. Summary and discussion

Our numerical results indicate that the choice of the logarithmic error function $E_L(\vec{W})$ over the quadratic error function $E_Q(\vec{W})$ results in an improved learning algorithm for layered neural networks. Such improvement can be explained in terms of the characteristic steepness of the landscape defined by each error function in configuration space $\{\vec{W}\}$. A quantitative measure of the average steepness of the $E(\vec{W})$ surface is given by $\mathcal{S} \equiv < |\vec{\nabla} E(\vec{W})| >$. Reliable estimates of $\mathcal{S}$ from random sampling in $\{\vec{W}\}$ space yield $\mathcal{S}_L/\mathcal{S}_Q \simeq 3/2$. The increased steepness of $E_L(\vec{W})$ over $E_Q(\vec{W})$ is in agreement with the schematic landscape shown in figure 5, and explains the reduction in learning time found in our numerical simulations.

A steeper error function $E_L(\vec{W})$ has been shown to lead to an improved learning algorithm when gradient descent is used to solve the $E(\vec{W}) = 0$ minimization problem. That the network design resulting from successful learning does not exhibit improved generalization abilities can be understood as follows. Consider the total error function

$$\overline{E} = \sum_{\text{all } \alpha} d^\alpha = \sum_{\alpha \in T} d^\alpha + \sum_{\alpha \notin T} d^\alpha. \tag{4.1}$$

The first term includes all patterns in the training set T, and is the error function E which is minimized by the learning algorithm. The second term is the error made by the network on all patterns not in the training set, and measures *generalization ability*. As shown in figure 6, equally valid solutions to the learning problem $E(\vec{W}) = 0$ vary widely in their generalization ability as measured by $\overline{E}(\vec{W})$. We argue that the application of increasingly sophisticated techniques to the learning problem posed as an unconstrained

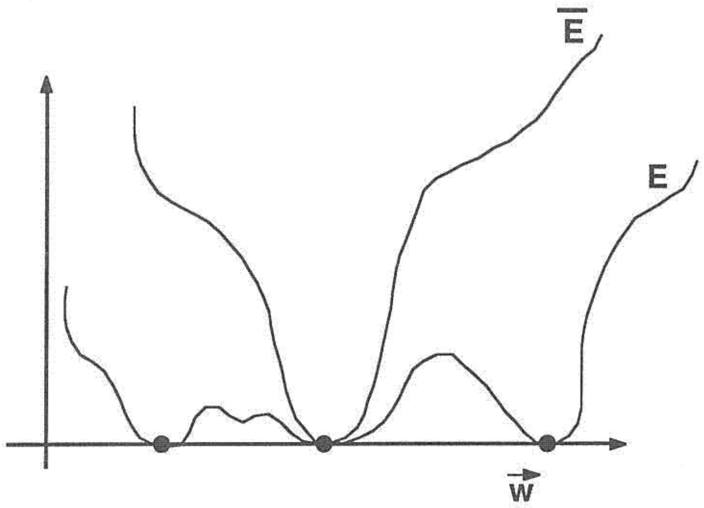


Figure 6: Valid solutions to the $E(\vec{W}) = 0$ learning problem, indicated by •, exhibit varying degrees of generalization ability as measured by $\overline{E}(\vec{W})$.

optimization problem does not lead to marked improvement in the generalization ability of the resulting network.

It is nevertheless desirable to use computationally efficient learning algorithms. For gradient descent learning, such efficiency is controlled by the steepness and density of local minima of the surface defined by $E(\vec{W})$ in configuration space. Our work shows that notable reductions in learning time can result from an appropriate choice of error function.

The question that remains open is whether the logarithmic error function is generally to be preferred over the quadratic error function. The accelerated learning found for the contiguity problem, and other test problems for which we have performed comparative simulations, has been explained here as a consequence of the smoothness and steepness of the surface $E(\vec{W})$. We know of no argument in support of $E_L(\vec{W})$ being always smoother than $E_Q(\vec{W})$; the smoothness of the $E(\vec{W})$ surface depends subtly on the problem, its representation, and the choice of training set T. As for the steepness, the argument presented in the Appendix proves that $E_L(\vec{W})$ is indeed steeper than $E_Q(\vec{W})$ in the vicinity of local minima, in support of the ubiquitous reduction in learning time.

Acknowledgments

We have enjoyed conversations on this subject with John Hopfield. We thank him for sharing his insight, and for making a copy of his paper available to us before publication.

Appendix

Consider the error function

$$E = \sum_{\alpha=1}^{m} \sum_{j=1}^{N_L} d_j^\alpha, \tag{A.1}$$

with

$$d_j^\alpha = \frac{1}{2} \, (\mathcal{O}_j^\alpha - \mathcal{T}_j^\alpha)^2 \tag{A.2}$$

for E_Q, and

$$d_j^\alpha = \mathcal{T}_j^\alpha \, \ln \frac{\mathcal{T}_j^\alpha}{\mathcal{O}_j^\alpha} + (1 - \mathcal{T}_j^\alpha) \, \ln \frac{(1 - \mathcal{T}_j^\alpha)}{(1 - \mathcal{O}_j^\alpha)} \tag{A.3}$$

for E_L.

The effect of a discrepancy $\Delta_j^\alpha = \mathcal{O}_j^\alpha - \mathcal{T}_j^\alpha$ between target and output on d_j^α is easily calculated, and can be written as

$$d_j^\alpha = \frac{1}{2} \, (\Delta_j^\alpha)^2 \tag{A.4}$$

for E_Q, and

$$d_j^\alpha = \frac{1}{2} \, A_L \, (\Delta_j^\alpha)^2 + \mathrm{O}((\Delta_j^\alpha)^3) \tag{A.5}$$

for E_L, $0 < \mathcal{T}_j^\alpha < 1$. Both error functions are quadratic to lowest order in Δ_j^α. But the parabolic trough that confines $\mathcal{O}_j^\alpha$ to the vicinity of $\mathcal{T}_j^\alpha$ is much steeper for E_L than for E_Q, since the function $A_L = 1/(\mathcal{T}_j^\alpha(1 - \mathcal{T}_j^\alpha))$ satisfies $A_L \geq 4$ in the interval $0 < \mathcal{T}_j^\alpha < 1$. This function reaches its minimum value $A_L = 4$ at $\mathcal{T}_j^\alpha = 1/2$, and diverges as $\mathcal{T}_j^\alpha$ approaches $0, 1$.

The quadratic expansion of equation (A.5) for E_L is not valid at $\mathcal{T}_j^\alpha = 0, 1$. At the ends of the interval the correct expansion is

$$d_j^\alpha = \pm \Delta_j^\alpha + \frac{1}{2} \, (\Delta_j^\alpha)^2 + \mathrm{O}((\Delta_j^\alpha)^3), \tag{A.6}$$

with the plus sign valid for $\mathcal{T}_j^\alpha = 0$, $\mathcal{O}_j^\alpha = \Delta_j^\alpha$, and the minus sign valid for $\mathcal{T}_j^\alpha = 1$, $\mathcal{O}_j^\alpha = 1 + \Delta_j^\alpha$. For $\mathcal{T}_j^\alpha = 0, 1$ the confining effect of E_L is again stronger than that of E_Q, due to the leading linear term.

The preceding analysis demonstrates that $E_L(\vec{\mathcal{O}}^\alpha)$ is steeper than $E_Q(\vec{\mathcal{O}}^\alpha)$ in the vicinity of the minima at $\vec{\mathcal{O}}^\alpha = \vec{\mathcal{T}}^\alpha$. Since both functions depend implicitly on the couplings $\{\vec{W}\}$ through the outputs $\{\vec{\mathcal{O}}^\alpha\}$, it follows that $E_L(\vec{W})$ is steeper than $E_Q(\vec{W})$. Note that the components of the gradient $\nabla E(\vec{W})$ are given by

$$\frac{\partial E}{\partial W_{ik}^{(\ell)}} = \sum_{\alpha=1}^{m} \sum_{j=1}^{N_L} \frac{\partial d_j^\alpha}{\partial \mathcal{O}_j^\alpha} \frac{\partial \mathcal{O}_j^\alpha}{\partial W_{ik}^{(\ell)}}. \tag{A.7}$$

The factors $\partial \mathcal{O}_j^\alpha / \partial W_{ik}^{(\ell)}$ depend only on the processing of information along the network, from layer ℓ to the top layer at $\ell = L$, and are independent of the choice of distance $d^\alpha = d(\vec{\mathcal{O}}^\alpha, \vec{\mathcal{T}}^\alpha)$. An increase in the magnitude of $\partial d_j^\alpha / \partial \mathcal{O}_j^\alpha$ in the vicinity of $\mathcal{O}_j^\alpha = \mathcal{T}_j^\alpha$ thus results in an increase in the magnitude of the gradient of $E(\vec{W})$.

References

[1] E. B. Baum and F. Wilczek, "Supervised Learning of Probability Distributions by Neural Networks", to appear in *Proceedings of the IEEE Conference on Neural Information Processing Systems - Natural and Synthetic*, (Denver, 1987).

[2] J. J. Hopfield, "Learning Algorithms and Probability Distributions in Feed-forward and Feed-back Networks", *Proc. Natl. Acad. Sci. USA*, **84** (1987) 8429-8433.

[3] G. E. Hinton, "Connectionist Learning Procedures", Technical Report CMU-CS-87-115.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation", in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, (MIT Press, 1986).

[5] T. J. Sejnowski and C. R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text", *Complex Systems*, **1** (1987) 145-168.

[6] A. J. Viterbi, *Principles of Digital Communication and Coding*, (McGraw-Hill, 1979).

[7] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, (Academic Press, 1972).

[8] J. S. Denker, D. B. Schwartz, B. S. Wittner, S. A. Solla, R. E. Howard, L. D. Jackel, and J. J. Hopfield, "Automatic Learning, Rule Extraction, and Generalization", *Complex Systems*, **1** (1987) 877-922.

[9] S. A. Solla, S. A. Ryckebusch, and D. B. Schwartz, "Learning and Generalization in Layered Neural Networks: the Contiguity Problem", to be published.

[10] T. Maxwell, C. L. Giles, and Y. C. Lee, "Generalization in Neural Networks: the Contiguity Problem", in *Proceedings of the IEEE First International Conference on Neural Networks*, (San Diego, 1987).

[11] T. Grossman, R. Meir, and E. Domany, "Learning by Choice of Internal Representations", to be published.