Learning in the multilayer perceptron

1988 J. Phys. A: Math. Gen. 21 2643

(http://iopscience.iop.org/0305-4470/21/11/021)

J. Phys. A: Math. Gen. **21** (1988) 2643-2650. Printed in the UK

Learning in the multilayer perceptron

Keith A Benedict

Department of Theoretical Physics, The Schuster Laboratory, The University, Manchester M13 9PL, UK

Received 14 December 1987, in final form 1 March 1988

Abstract. Learning in a generalised perceptron neural network model is investigated by numerical simulation. It is found that the distribution of learning times is very broad and spreads out as the system size is increased. The mean number of steps, k, to learn a first-order task is found to increase with system size according to a power law $\langle k \rangle \sim N^{\alpha}$, $\alpha = 1.86 \pm 0.05$.

1. Introduction

The recent growth of interest in neural network models in the physics community was prompted by the discovery of Hopfield (1982) that a highly connected system of formal neurons, similar to the Sherrington-Kirkpàtrick (1975) model, could act as a content-addressable memory; a system which can recall a previously stored pattern when presented with a corrupted or incomplete version of that pattern. Such a system clearly has great potential in the field of image recognition. It is, however, hoped that neural nets can provide both a model for neurophysiological processes and an alternative to conventional artificial intelligence. The complexity of tasks which the Hopfield model can learn to do by example is believed to be limited to those which are first order in the classification of Minsky and Papert (1969); attention has therefore shifted toward other neural net models with greater scope. One such model is a generalisation of the perceptron of Rosenblatt (1962) due to the PDP group (Rummelhart and McLelland 1986), which has extra, hidden (i.e. neither input nor output) neurons; this is known as the multilayer perceptron or feed-forward net.

A learning algorithm for this network was proposed by Rummelhart *et al* (1985) and Le Cun (1985), called back-propagation, in which a cost function, proportional to the total number of errors made by the net, is minimised by a modified gradient descent.

It is clearly of importance to know how the typical number of steps, k, needed to learn a task varies with the parameters in the problem. Valiant (1984) and Volper and Hanson (1986) have proposed that k varies exponentially with the order of the task being learnt. Tesauro (1987) has considered the variation of k with the number of patterns, ν, in the training set for the 32-bit parity problem and arrives at the relation

$$k \sim \nu^{4/3}.$$

Rummelhart *et al* consider the variation of k with the number of hidden units, m, and find

$$k \sim -\log m.$$

0305-4470/88/112643+08$02.50

The work described here addresses the question of the variation of k with the number of bits in the input word, specifically for the content addressable memory task. Scalettar and Zee (1986) considered this and performed simulations in which the number of patterns and the number of hidden units were fixed: only the word length, n, was varied. They concluded that k tended to a constant as n increased. Here we shall consider the case in which n, m and ν are scaled up together and propose a relation of the form

$$k \sim n^{\alpha} \qquad \nu = m = [\tfrac{1}{2}n]$$

for this particular task.

The learning time of this model for CAM is not of interest in itself because the original perception with its learning rule can learn the task much faster than the multilayer perception using back-propagation. The original model, however, cannot do the higher-order tasks at all. The only reason that CAM has been investigated is that back-propagation is so much slower for the more complex tasks that the collection of a sufficiently large amount of data would require excessive computer time.

The remainder of this paper is arranged as follows: § 2 describes the feed-forward net and the back-propagation algorithm, § 3 discusses the numerical simulations, § 4 gives their results and § 5 contains a brief summary and a few comments on the results.

2. The three-layer perceptron and back-propagation

The neural net studied here has the architecture shown in figure 1. Each neuron is labelled by which layer it is in ($\alpha = 0, 1, 2$) and by its location within the layer ($i = 0, 1, \ldots, n^{(\alpha)}$). The state of each neuron is a continuous variable $S_i^{\alpha} \in [-1, 1]$ except for the end neuron in each layer which is fixed to always have the value $S_0^{\alpha} = -1$ (this plays the role of a source of threshold potentials). The bond between neurons (α, i) and $(\alpha - 1, j)$ has an associated weight J_{ij}^{α}. The input to the net is a binary word: $I_i \in \{-1, 1\}$ $i = 1, \ldots, n^{(0)}$. The state of the ith neuron in the lower layer is set to the

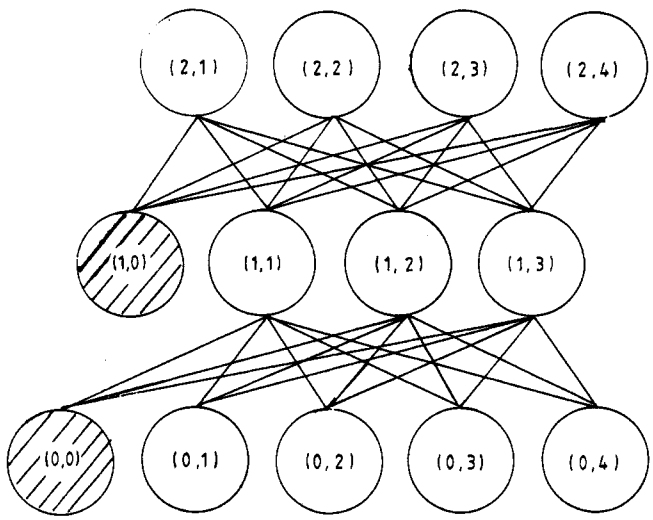

Figure 1. The architecture of the three-layer perceptron. Open circles represent dynamical neurons. Shaded circles are the quenched neurons.

value I_i. The states in the middle layer are set by a dynamical rule and then so are the states in the upper layer. The rule is the following:

$$S_i^\alpha = f_\beta\left(\sum_{j=1}^{n^{(\alpha-1)}} J_{ij}^\alpha S_j^{\alpha-1}\right) \qquad f_\beta(x) = \tanh(\tfrac{1}{2}\beta x).$$

The states in the upper layer are then taken as the output of the net; this output is only a binary word if $\beta = \infty$ (i.e. $\tanh(\frac{1}{2}\beta x) = \mathrm{sgn}(x)$). This 'feed-forward' dynamics is strictly deterministic and, for fixed weights, a given input $\boldsymbol{I}$ will elicit a unique output $\boldsymbol{O}(\beta, \mathbf{J}, \boldsymbol{I})$.

Learning is thus a question of finding a set of weights, $\mathbf{J}$, which correctly associates each word from an input 'training set' with the corresponding word from an output set in the limit $\beta \to \infty$.

Let the input training set be $\{\boldsymbol{I}^{(p)}: p = 1, \ldots, \nu\}$ and the output set be $\{\boldsymbol{\tau}^{(p)}\}$. The back-propagation algorithm defines an energy or cost function,

$$E_\beta(\mathbf{J}) = \tfrac{1}{2}\sum_{p=1}^{\nu}\sum_{i=1}^{n^{(p)}} |O_i(\beta, \mathbf{J}, \boldsymbol{I}^{(p)}) - \tau_i^{(p)}|^2.$$

The weights $\mathbf{J}$ are initially chosen at random and then iteratively adjusted according to the rule

$$\mathbf{J}(k+1) = \mathbf{J}(k) - \eta \nabla_{\mathbf{J}} E_\beta(\mathbf{J}(k)) + \mu(\mathbf{J}(k) - \mathbf{J}(k-1)).$$

A useful analogy is to a particle moving on a complex, high-dimensional surface. The gradient term is a 'gravitational' force pulling the particle downhill while the second term is an 'inertial' force or momentum tending to keep the particle moving in the same direction. The introduction of momentum speeds up the convergence of the algorithm and allows the particle to escape from narrow local minima.

For a given training set, initial weight vector $\boldsymbol{J}(0)$ and set of parameters ($n^{(\alpha)}$, η, μ and β) there is a unique number of steps, $k(\boldsymbol{J}(0))$ for the descent to find a solution (not necessarily finite because of broad local minima). We wish to know, for a given architecture $\{n^{(\alpha)}\}$ and fixed size of training set, ν, the distribution of k ($\mathbf{J}(0)$, $\{\boldsymbol{I}\}$, $\{\tau\}$) over all starting weights and all such training sets. We define this to be $\rho(k)$, $k = 0, 1, \ldots, \infty$, satisfying

$$\sum_{k \geq 0} \rho(k) = 1.$$

Clearly the computational time for the calculation of one step scales with wordlength, n, according to the number of bonds, i.e. $\sim n^2$ so the mean learning time will scale as

$$\langle t \rangle \sim n^{\alpha+2}.$$

For the CAM task discussed in the rest of this paper $n^{(0)} = n^{(2)} = n$, $n^{(1)} = m$ and the input and output sets are identical.

3. Numerical simulations of learning

In the simulations β was fixed at one during the gradient descent, but the criterion for termination was that $E_\infty(\mathbf{J}) = 0$, i.e. the output matched the input exactly, with $f(x) = \mathrm{sgn}\ (x)$. An upper limit of $100n^2$ steps was imposed: any descent not having converged after this many steps was restarted at a new, random point in weight space and the step counter set to zero; in a typical run of 5000 descents this occurred less than 20 times and is due in these cases to particularly difficult training sets containing

two patterns differing in only one bit (for higher-order tasks failure to converge is a much more serious problem, even when the step limit is greatly increased, indicating the importance of broad local minima in these more complex tasks). Each weight, J_{ij}^{α}, was chosen with uniform probability from the interval $[-5, 5]$.

Each run consisted of N descents, $g = 1, \ldots, N$: the number of steps taken in the descent k_g was recorded. The distribution $\rho(k)$ approximated by

$$\tilde{\rho}(k) = \frac{1}{N} \sum_{g=1}^{N} \delta(k, k_g)$$

and the mean value by

$$\langle k \rangle = \bar{k} \pm \frac{1}{\sqrt{N}} \sqrt{(\overline{k^2})} \qquad \overline{k^l} = \frac{1}{N} \sum_{g} (k_g)^l.$$

In the first set of runs n, m and ν were fixed and η and μ were varied between 0.1 and 0.9 in 0.1 steps. The values which minimised $\bar{k}$ were then used in subsequent simulations. In the next set of runs n was fixed and the number of patterns, ν, and the number of hidden units, m, varied together (i.e. $m = \nu$). This showed that for $\nu \geqslant \frac{1}{2}n$ the mean learning time was constant within errors. Consequently in the main simulations the training set size and the number of hidden units were fixed at $\nu = m = [\frac{1}{2}n]$. The main simulations consisted of 1200 descents for each value of n in the range $n = 3, 4, \ldots, 12$.

4. Results

Figures 2 and 3 show graphs of $\bar{k}$ against η and μ from the first simulations with $n = 8$, $m = 4$, $\nu = 4$ from 500 runs per point. In subsequent simulations $\eta = 0.4$ and $\mu = 0.5$ were chosen.

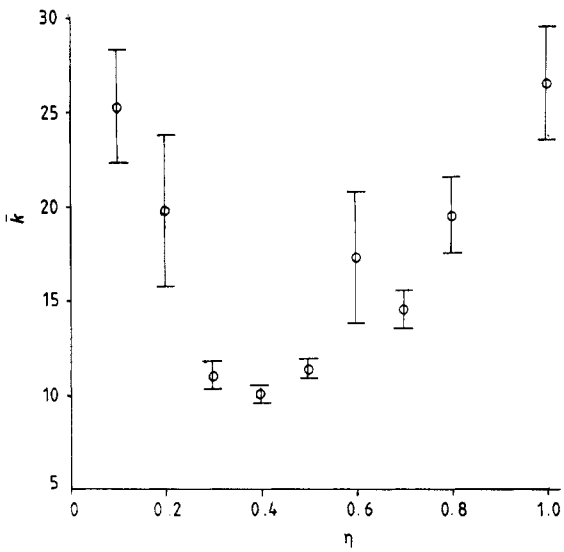

Figure 2. Plot of mean learning time against the 'downhill' step parameter, η.

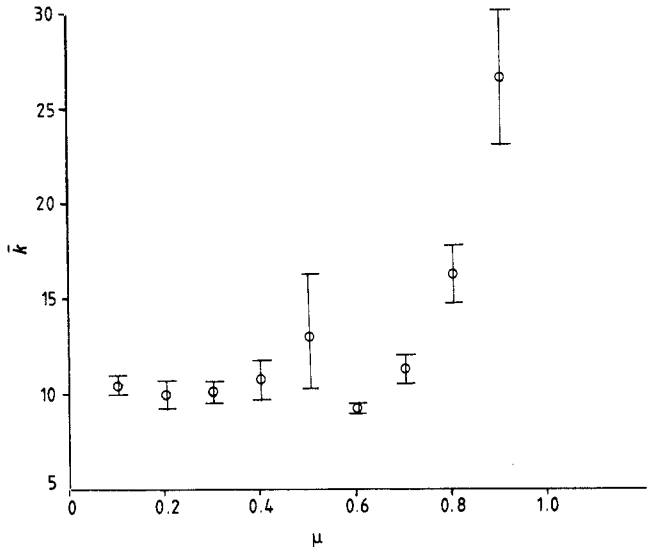

Figure 3. Plot of mean learning time against the momentum parameter, μ.

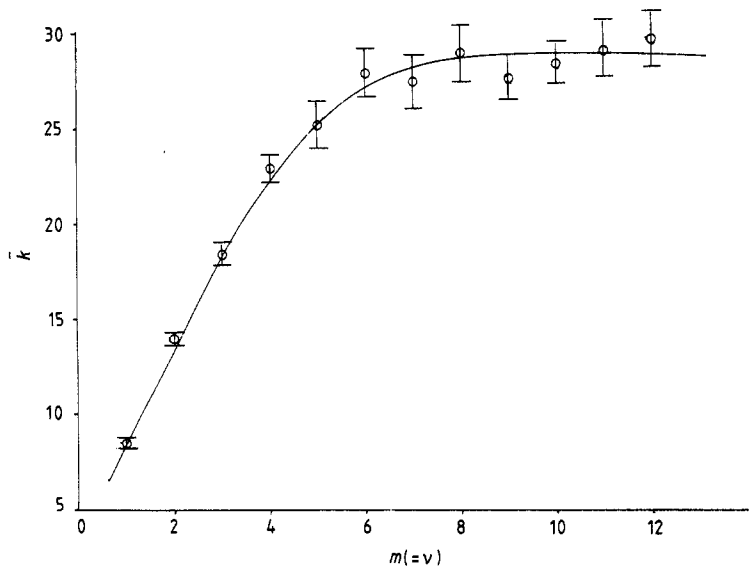

Figure 4. Plot of mean learning time against number of patterns learnt, with the number of hidden units, m, always equal to the number of patterns.

Figure 4 shows a graph of $\bar{k}$ against $m(=\nu)$ for $n=8$ with 500 runs per point. Clearly for $m \geqslant \frac{1}{2}n$ this is effectively constant.

Figure 5 shows a graph of $\log \bar{k}$ against $\log n$ for $n=3, 4, \ldots, 12$. The best fit line to this set of points has slope 1.8 ± 0.2 and we therefore infer the scaling law

$$\langle k \rangle \sim n^{\alpha} \qquad \alpha = 1.86 \pm 0.05.$$

Figures 6 and 7 show the distribution $\tilde{\rho}(k)$ for $n=4$ and $n=12$. Clearly for the larger value of n the distribution has spread out, in fact the width appears to scale

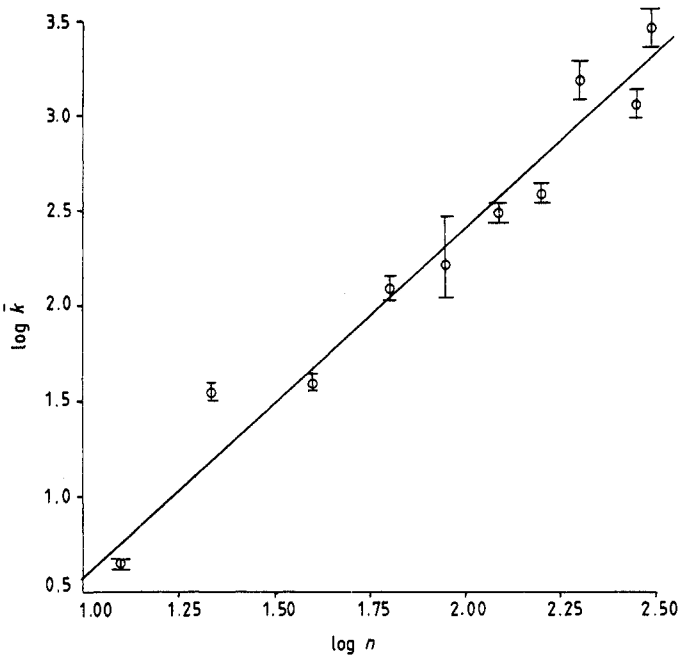

Figure 5. A log-log plot of mean learning time against word length.

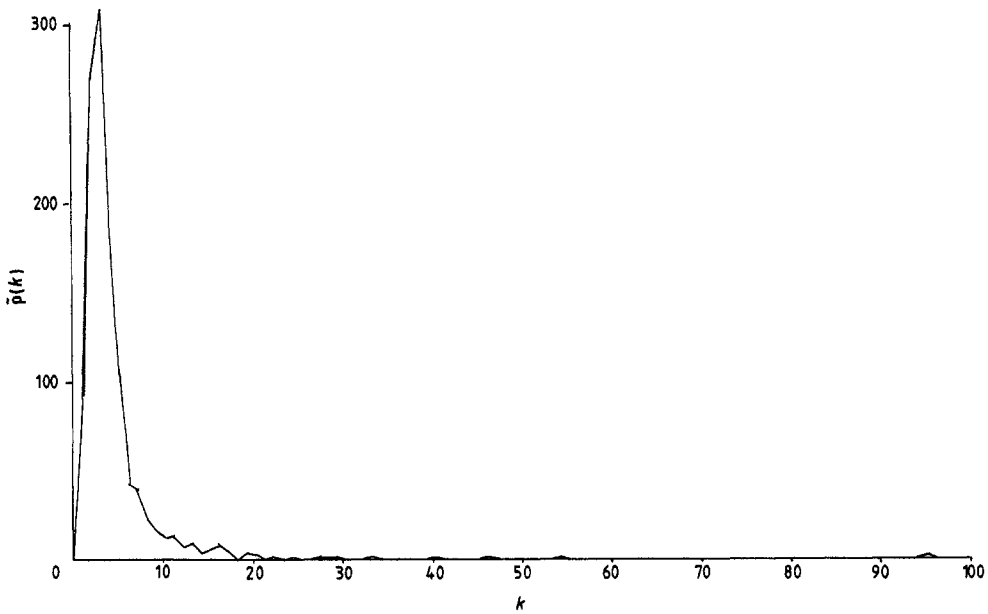

Figure 6. The distribution of learning times $\tilde{\rho}^{(k)}$ for $n=4$ and $m=\nu=2$.

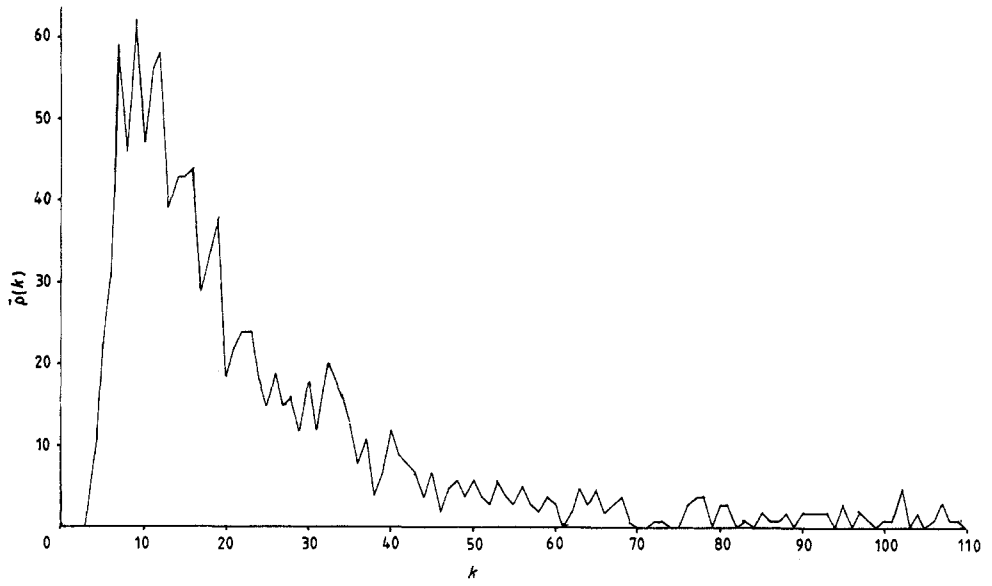

Figure 7. The distribution of learning times $\tilde{\rho}^{(k)}$ for $n = 12$ and $m = \nu = 6$.

approximately as

$$W = \frac{\overline{k^2}}{(\bar{k})^2} \sim n^2$$

although a precise determination of the exponent would require much more data.

5. Comments

The errors shown on the graphs are derived from the second moment of the distribution $\tilde{\rho}(k)$. Because of the breadth of this distribution they do not represent the usual levels of confidence. For example the value $\alpha = 2$ is not impossible, but there is no particular reason why it should be an integer, and seems fairly unlikely from the data.

The fact that the learning time does not show any tendency to self-average, as observed by Smeija and Richards (1987) (i.e. W does not vanish as $n \to \infty$), is not so surprising. Consider the following problem, which is in no way to be taken as a model of back-propagation but may have some features in common. A particle executes a random walk on the edges of a d-dimensional tehrahedron: the walk starts at a randomly chosen vertex and terminates when the particle lands on one special 'target' site. The probability that this random search takes l steps is

$$P_l = \frac{1}{n}\left(1 - \frac{1}{n}\right)^l \sim \frac{1}{n} e^{-l/n} \qquad n(=d+1) \to \infty.$$

The mean number of steps is then $\bar{l} = n$ while $\overline{l^2} = 2n^2$ hence $W \sim$ constant as $n \to \infty$ so that the distribution of search times does not self-average even in this very simple example.

In summary it has been shown that the mean number of steps required in learning by back-propagation, for a first-order task, scales with word length according to $\langle k \rangle \sim n^{1.86}$, but that the distribution of learning times is very broad.

It should, perhaps, be stressed again that this result is specific to the task examined (CAM). The exponent α will undoubtedly be much larger (if it is definable at all) for harder learning tasks. If there is any universality in the learning in these systems then there will be many universality classes labelled by the architectural parameters of the network, the Minsky–Papert order of the task and other measures of learning difficulty.

Acknowledgments

It is a pleasure to thank M A Moore and J S Shapiro for many useful and stimulating discussions and the SERC for their financial support.

References

Hopfield J J 1982 *Proc. Natl Acad. Sci. USA* **79** 2554
Le Cun Y 1985 *Proc. Cognitiva* **85** 599
Minsky M and Papert S 1969 *Perceptrons* (Cambridge MA: MIT Press)
Rosenblatt F 1962 *Principles of Neurodynamics* (New York: Spartan)
Rummelhart D E, Hinton G E and Williams R J 1985 *UCSD Internal Report*
Rummelhart D E and McLelland J L 1986 *Parallel Distributed Processing* (Cambridge, MA: MIT Press)
Scalettar R and Zee A 1986 *Santa Barbara Preprint* NSF-ITP 86-118
Sherrington D and Kirkpatrick S 1975 *Phys. Rev. Lett.* **32** 1742
Smeija F and Richards G 1987 *Edinburgh University Preprint* 87/418
Tesauro G 1987 *Complex Systems* **1** 367
Valiant 1984 *Commun. ACM* **27** 1134
Volper and Hanson 1986 *Biol. Cybernet.* **54** 393