Network: Computation in Neural Systems **7** (1996) 277–284. Printed in the UK

WORKSHOP PAPER

Development of low entropy coding in a recurrent network*

George F Harpur and Richard W Prager

Engineering Department, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK

Received 1 February 1996

Abstract. In this paper we present an unsupervised neural network which exhibits competition between units via inhibitory feedback. The operation is such as to minimize reconstruction error, both for individual patterns, and over the entire training set. A key difference from networks which perform principal components analysis, or one of its variants, is the ability to converge to non-orthogonal weight values. We discuss the network's operation in relation to the twin goals of maximizing information transfer and minimizing code entropy, and show how the assignment of prior probabilities to network outputs can help to reduce entropy. We present results from two binary coding problems, and from experiments with image coding.

1. Introduction

The problem of unsupervised learning can be seen as one of 'automatic coding,' or attempting to infer from a set of input patterns a new representation of those patterns which (i) transfers all relevant information and (ii) does so efficiently. The idea of *relevant* information is task-specific, however, so a general purpose unsupervised system should in fact aim to transfer *all* input information, with only minimal assumptions about the nature of noise within the input signal.

In this paper, we explore the use of two information-theoretic goals to achieve these aims:

(i) *Maximizing information transfer (Linsker 1988).* The mutual information between input and output is maximized when there is a unique mapping from the output back to the input, since it must be the case that if the input can be completely reconstructed from the output, then the output carries all the input information. Where such a mapping is not achievable, we aim to minimize the sum-squared reconstruction error. This choice of error metric can be justified in both information-theoretic and maximum-likelihood frameworks (Plumbley 1991, Harpur and Prager 1995).

(ii) *Minimizing code entropy (Barlow 1989).* Goal (i) above pushes up the lower bound on the total entropy of the output units. By minimizing code entropy, the network attempts to approach this bound as closely as possible from above and hence to produce the most efficient code that captures all input information. The code entropy will be at its minimum when there is no mutual information between output units; that is, when all outputs are statistically independent. Under the assumption that the underlying causes

* This paper was presented at the Workshop on Information Theory and the Brain, held at the University of Stirling, UK, on 4–5 September 1995.

0954-898X/96/020277+08$19.50

of the data are themselves independent, each output unit will represent one of those causes, and so as well as producing an efficient coding, the network will in some sense have 'understood' the input data.

2. Network architecture

The basic network architecture used is shown in figure 1. The network has m inputs and n outputs. The input vector $\boldsymbol{x} = (x_1, \ldots, x_m)^{\mathrm{T}}$ (where $^{\mathrm{T}}$ denotes the transpose) is fed to a first layer of linear units which form the sum of the input and the feedback from the second layer outputs $\boldsymbol{a} = (a_1, \ldots, a_n)^{\mathrm{T}}$. The excitatory feedforward connections between the first and second layers are mirrored by inhibitory feedback connections with weights of the same magnitude but opposite sign. The feedforward weights to the ith unit in the second layer are denoted by the vector $\boldsymbol{w}_i = (w_{i1}, \ldots, w_{im})^{\mathrm{T}}$. The feedback weights *from* the same unit are therefore simply $-\boldsymbol{w}_i$, and so the total inhibitory feedback is $\boldsymbol{y} = \sum_{k=1}^{n} a_k \boldsymbol{w}_k$, giving $\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{y}$ as the net output from the first layer. The outputs, initially zero, are calculated iteratively using the rule $\Delta a_i = \mu \boldsymbol{x}' \cdot \boldsymbol{w}_i$ where μ is an adjustable rate ($0 < \mu \leqslant 1$). The effect of the feedback is that when an output unit is activated, it 'claims' the part of the input that it has responded to by subtracting it from $\boldsymbol{x}$, thus preventing any other unit from responding to the same feature. In this way the output units compete for activation, but without preventing more than one unit from being activated simultaneously when input patterns from multiple causes are superimposed. It can easily be shown (Harpur and Prager 1995) that at equilibrium this activation scheme minimizes the sum-squared reconstruction error, $E = |\boldsymbol{x}'|^2$. This in turn is equivalent to maximizing information transfer under the assumption of independent spherical Gaussian input noise (Plumbley 1991).

The weights are adapted after activation using local Hebbian learning between first and second layer units: $\Delta \boldsymbol{w}_i = \eta a_i \boldsymbol{x}'$, where η is the learning rate. Harpur and Prager (1995) show that this rule performs gradient descent on the reconstruction error as the activation rule

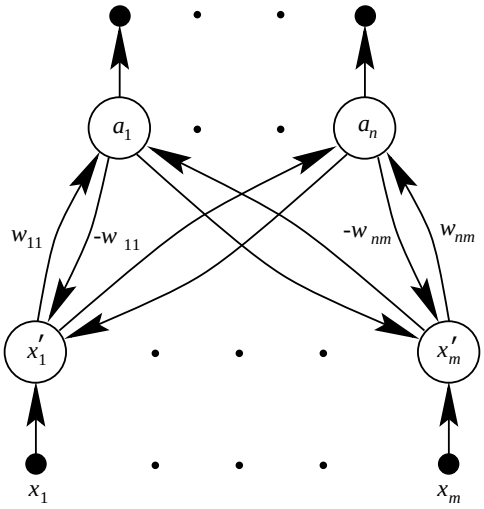


Figure 1. A two-layer network architecture: competition is achieved via feedback connections, whose weight values have the opposite sign to the corresponding feedforward weight. The symbols within the circles represent the output of that unit.

did, but now averaged over all the patterns presented to the network. Note that the network algorithm bears a resemblance to a number of principal component analysing networks (e.g. Williams 1985, Sanger 1989). However, such networks omit the recurrent activation stage (and therefore the competition between units), and are thus forced to converge to a set of *orthogonal* weight vectors, so that while pairs of outputs are decorrelated, they cannot in general be statistically independent, since there is no reason to suppose that the underlying causes of the data form a mutually orthogonal set.

3. Penalty terms and prior probabilities

The algorithm described in section 2 searches for a set of vectors which can additively encode the input patterns with minimum mean square error. When there are less outputs than inputs ($n < m$), this may well produce interesting and useful encodings of the input patterns. If we set $n \geqslant m$, however, the search is under-constrained since *any* m weight vectors (providing they are linearly independent) will fully span the input space, and hence be able to encode any input pattern with zero error.

We have already stated that in this situation, the secondary aim of the network is to further constrain the solution by attempting to squeeze out redundancy by minimizing code entropy. Unfortunately, entropy itself is difficult to work with, firstly because it does not generalize well from the discrete to the continuous case as discussed, for example, by Cover and Thomas (1991). Secondly, although low entropy distributions have the common feature of being highly peaked at one or more locations, working with entropy values alone would not give any clue as to the whereabouts of these locations, and so would not usefully constrain the learning process.

A more useful technique is to work with probability distributions directly. If we can produce reasonable approximations to the probability distributions of the underlying causes of the data, and can cajole the output values into reproducing these distributions (while still maximizing information transfer), then the weight values should more closely resemble those causes, and the code entropy will be reduced.

One important example of incorporating such prior knowledge is in constraining output values to be non-negative. When considering feature-detecting networks, it is natural to talk about a real-world feature being present to a greater or lesser degree, but it is usually nonsensical to assign a negative value to its presence in the input. Building this fact into the network is simply a matter of using threshold linear output units (with the threshold at zero), but this nonlinearity has a dramatic effect on the operation of the network (Harpur and Prager 1995).

Similarly, when dealing with a domain that is essentially binary in nature, it is useful to assume a prior probability distribution for output values that is peaked at zero and one.

Several authors (e.g. Barlow 1989, Földiák 1992, Field 1994), have identified sparse coding as a useful goal for unsupervised learning. For any particular input, only a small number of outputs have non-zero values. We can make this idea more concrete by saying that the aim is for each output unit to have an (independent) output probability distribution sharply peaked at zero. At the same time it must have sufficient variance to enable the primary aim of capturing all input information to be achieved. This idea has led to Field (1994) identifying the kurtosis (heaviness of the tail) of the distribution as being important, since for such symmetrical distributions of fixed variance, the greater the kurtosis the lower the entropy. The notion of searching for maximally non-Gaussian distributions (the Gaussian has zero kurtosis and *maximum* entropy for a fixed variance) is also to be found in the literature on exploratory projection pursuit (Huber 1985).

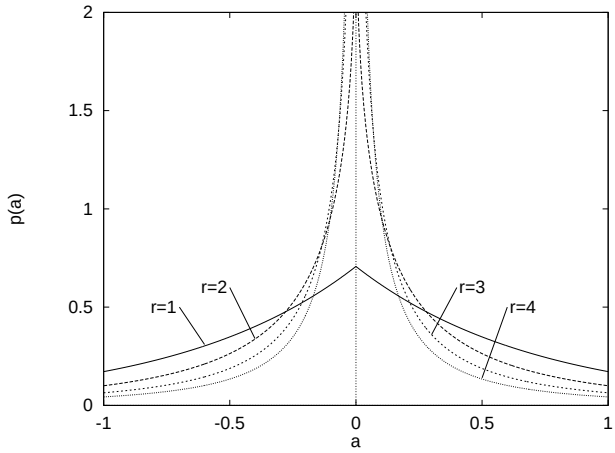


Figure 2. Examples from the family of probability density functions (1) for $r = 1, 2, 3, 4$. All have been normalized to have unit variance.

We consider here the family of distributions whose probability densities are given by

$$p(a) = \frac{1}{2\beta r!}\ \exp\left[-\left(\frac{|a|}{\beta}\right)^{1/r}\right] \tag{1}$$

for some integer $r \geqslant 1$. These can be normalized to have unit variance by setting

$$\beta = \left[\frac{(r-1)!}{(3r-1)!}\right]^{1/2}.$$

The distributions are shown for the first four values of r in figure 2, and the corresponding kurtoses and entropies are listed in table 1. The kurtoses were calculated by symbolic integration of $\int_{-\infty}^{\infty} a^4 p(a)\,\mathrm{d}a$, and the entropies using $-\int_{-\infty}^{\infty} p(a)\log[p(a)]\,\mathrm{d}a$.

Table 1. Values of kurtosis and entropy for the probability density functions shown in figure 2.

r	Kurtosis	Entropy
1	6.0	1.347
2	25.2	0.993
3	107.3	0.529
4	458.1	0.016

A probabilistic analysis of neural networks (e.g. Mackay 1995) tells us that the use of a sum-squared error term can be interpreted as performing maximum-likelihood estimation under the assumption of an independent spherical Gaussian noise model (by taking minus the log-likelihood). Applying such an analysis to the network here tells us that the activation process determines the maximum-likelihood output values under the assumptions of the model given by the current weight values, Gaussian input noise and uniform prior probabilities for all possible output values. In order to incorporate the non-uniform priors shown in figure 2, we simply need to 'penalize' each output according to the minus logarithm of the probability density function, giving as the penalties $\pi(a) = |a|^{1/r}$. These penalties are incorporated into the objective function, giving $E_{\mathrm{p}} = \alpha|\boldsymbol{x}'|^2 + \sum_{i=1}^{n}\pi(a_i)$, with the

parameter α controlling the balance between the weight given to reconstruction error and activation penalties. In practice, when minimizing this function directly using gradient-based optimizers, it may prove necessary to add a small constant to the a_i before applying π to avoid the infinite gradient at zero for $r > 1$.

4. Experiments

The techniques described here have been applied to several problems, in both the binary and continuous domains. In these simulations the network was activated by direct minimization of the objective function E_p using a variable metric method, and weight values were initialized with random values and updated using the Hebbian learning rule of section 2. Further experimental detail may be found in Harpur and Prager (1995).

4.1. The lines data set

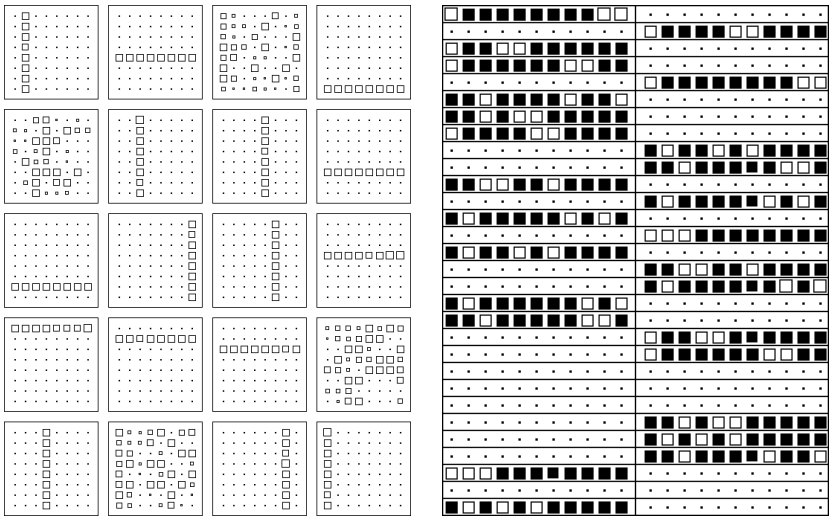
This experiment was introduced by Földiák (1992), and has subsequently been used by Zemel (1993) and Saund (1995). Input consists of an 8×8 grid, on which are placed combinations of the 16 possible horizontal and vertical lines, each one appearing with an independent probability p. At a point on the grid where there are one or more lines, the input is one; elsewhere it is zero. The outputs were constrained to be non-negative. Results for a data set with $p = 0.25$, giving an expected value of four superimposed lines per image, are shown in figure 3(*a*). Twenty outputs were used, of which 16 have identified the independent lines in the input, while the remainder are unused in the final solution. These results were typically reached after about 2000 pattern presentations to the unmodified network, but this figure was significantly reduced by including the term $\pi(a) = a^2 + (a-1)^2$ in the activation minimization which penalizes units with output values other than zero and one.

Note that the additive 'mixture model' used by the network is not correct in this example, since the input value where two lines intersect is one, and not two as the network would predict. Providing that p is kept reasonably small (less than about 0.4) this effect will not prevent the network from reaching the solution. See Saund (1995) for discussion of a 'soft-OR' mixture model which is more suited to this problem, and hence enables a solution to be reached with higher values of p.

4.2. The dual components data set

The second experiment used a data set described by Zemel (1993). There are 22 inputs. Patterns are produced by two independent processes, one producing the first 11 inputs, and the other the remaining 11. Each process produces one of 13 binary patterns (with equal probability). The mixture model is changed by encoding input binary zeroes as '-1', thus requiring such inputs to be 'claimed' explicitly by a unit. A weight value of 0 means that a unit is ambivalent as to the value of the corresponding input. The outputs were constrained to be non-negative. Good results (convergence within about 1400 pattern presentations) were obtained with a combination of the 'binary' penalty used in section 4.1 above and a linear 'sparseness' penalty corresponding to setting $r = 1$ in (1).

The resulting weights for a network of 30 outputs are shown in figure 3(*b*). The two halves have been separated for clarity, but note that this structure was not in any way present in the network itself—the training process alone revealed the division of the data into two independent halves.



(*a*) (*b*)

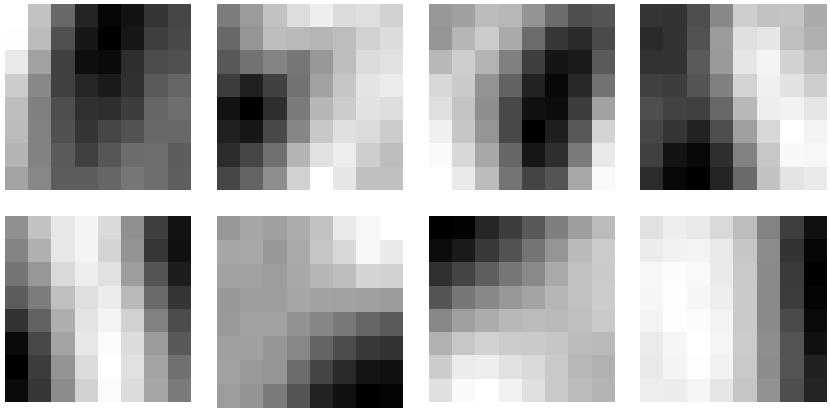
Figure 3. Final weight values from simulations using (*a*) the lines data set, and (*b*) the dual components data set. A clear box represents a positive weight value, and a filled box a negative value, with the length of the side proportional to its magnitude. Zero weights are indicated with points. In (*a*) each of the large squares represents the set of weights for one of the 20 output units. In (*b*) each of the 30 rows represents a single output unit; the vertical line indicates the bilateral structure present in the data (but not the network).

Note that this is not quite a minimum entropy code, since a '1' output from a unit on either side implies that all other units on that side will have output '0,' and hence the outputs are not statistically independent. Nevertheless, it does achieve the minimum entropy possible within the constraints of the reconstruction model.

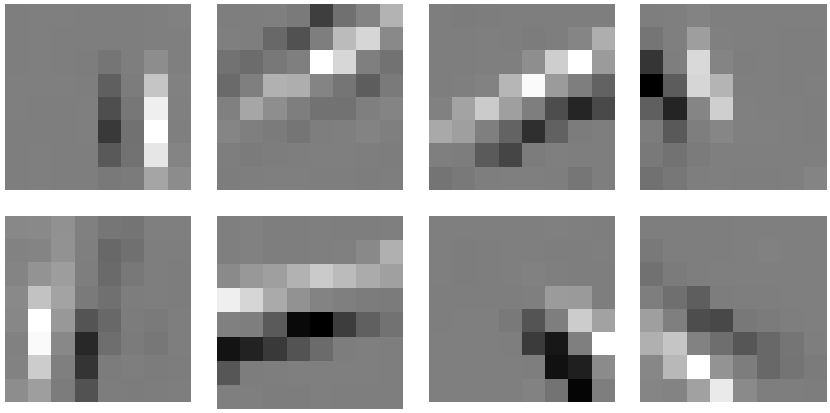
4.3. Image coding

A number of experiments in image coding have been performed. In each, a fully connected network initialized with random weights was trained on 8×8 patches of greyscale images. The first used a 256×256 pixel version of the well-known 'Lena' image and a network of just eight real-valued outputs, thus forcing a highly compressed representation. A linear penalty, corresponding to setting $r = 1$ in (1), was used. The mature weight values ('receptive fields') are shown in figure 4(*a*). When the image code was quantized to eight bits the resulting reconstruction error came to within 1% of that given by the first eight principal components (which give the theoretical minimum), but produced a 5% lower code entropy.

In a second set of experiments, following the work of Olshausen and Field (1996), a 64-output network was trained on a set of ten 512×512 natural images (without preprocessing) using a square-root penalty term ($r = 2$). Eight examples of the resulting weight vectors are shown in figure 4. These bear a good resemblance to the localized oriented wavelets that have been proposed by a number of authors (e.g. Daugman 1988, Field 1994) as the basis for performing sparse coding of visual scenes.



(*a*)



(*b*)

Figure 4. Examples of weight values produced by networks: (*a*) trained on the 'Lena' image with 8 output units, and (*b*) trained on natural scenes with 64 output units. Zero weights are depicted as mid-grey; positive weights are lighter, negative weights darker.

5. Discussion

The results obtained suggest that the network presented here, when combined with penalty terms to induce appropriate probability distributions on output values, is effective in discovering something of the underlying structure of its input data, and thus in producing low entropy codes. Future work will apply similar techniques to areas where the signal statistics are less well understood, such as artificial or medical images, with the aim of discovering useful ways of encoding such data for subsequent processing.

Acknowledgments

George Harpur is supported by a UK EPSRC postgraduate studentship. Many thanks are due to Bruno Olshausen and David Field for supplying test images, and to Bruno, John Daugman and Mark Plumbley for useful discussions and insights.

References

Barlow H B 1989 *Neural Comput.* **1** 295–311
Cover T M and Thomas J A 1991 *Elements of Information Theory* (New York: Wiley) ch 9
Daugman J G 1988 *IEEE Trans. Acoust. Speech Signal Process.* **36** 1169–79
Field D J 1994 *Neural Comput.* **6** 559–601
Földiák P 1992 *Models of Sensory Coding* Technical Report CUED/F-INFENG/TR 91, Engineering Department, Cambridge University, UK
Harpur G F and Prager R W 1995 *Techniques for Low Entropy Coding* Technical Report CUED/F-INFENG/TR 197, Engineering Department, Cambridge University, UK
Huber P J 1985 *Ann. Statist.* **13** 435–75
Linsker R 1988 *Computer* **21** 105–17
Mackay D 1995 *Network: Comput. Neural Syst.* **6** 469–505
Olshausen B J and Field D J 1996 *Network: Comput. Neural Syst.* **7** 333-9
Plumbley M 1991 On information theory and unsupervised neural networks *PhD Thesis* Engineering Department, Cambridge University, UK
Sanger T D 1989 *Neural Networks* **2** 459–73
Saund E 1995 *Neural Comput.* **7** 51–71
Williams R J 1985 *Feature Discovery Through Error-Correction Learning* Technical Report 8501, Institute for Cognitive Science, University of California
Zemel R S 1993 A minimum description length framework for unsupervised learning *PhD Thesis* Graduate Department of Computer Science, University of Toronto