Biol. Cybern. 64, 165–170 (1990)

Biological
Cybernetics

Forming sparse representations by local anti-Hebbian learning

P. Földiák

Physiological Laboratory, University of Cambridge, Downing Street, Cambridge CB2 3EG, United Kingdom

Received February 14, 1990/Accepted in revised form July 25, 1990

Abstract. How does the brain form a useful representation of its environment? It is shown here that a layer of simple Hebbian units connected by modifiable anti-Hebbian feed-back connections can learn to code a set of patterns in such a way that statistical dependency between the elements of the representation is reduced, while information is preserved. The resulting code is sparse, which is favourable if it is to be used as input to a subsequent supervised associative layer. The operation of the network is demonstrated on two simple problems.

1 Introduction

The brain receives a constantly changing array of signals from millions of receptor cells, but what we experience and what we are interested in are the objects in the environment that these signals carry information about. How do we make sense of a particular input when the number of possible patterns is so large that we are very unlikely to ever experience the same pattern twice? How do we transform these high dimensional patterns into symbolic representations that form an important part of our internal model of the environment? According to Barlow (1985) objects (and also features, concepts or anything that deserves a name) are collections of highly correlated properties. For instance, the properties 'furry', 'shorter than a metre', 'has tail', 'moves', 'animal', 'barks', etc. are highly correlated, i.e. the combination of these properties is much more frequent than it would be if they were independent (the probability of the conjunction is higher than the product of individual probabilities of the component features). It is these non-independent, redundant features, the 'suspicious coincidences' that define objects, features, concepts, categories, and these are what we should be detecting. While components of objects can be highly correlated, objects are relatively independent of one another. Subpatterns that are very highly correlated, e.g. the right- and left-hand sides of faces, are usually not considered as separate objects. Objects could therefore be defined as conjunctions of highly correlated sets of components that are relatively independent from other such conjunctions. The goal of the sensory system might be to detect these redundant features and to form a representation in which these redundancies are reduced and the independent features and objects are represented explicitly (Barlow 1961, 1972; Watanabe 1960, 1958).

2 Unsupervised learning

Learning in general is the process of the formation of a mapping from examples. Methods of supervised learning require either a 'teacher' that provides for each input the desired output or a reinforcer that reports whether the output generated was appropriate or not. These methods usually require a very large number of labelled examples. This is in sharp contrast with the ability of animals and people to learn from single or a relatively small number of examples, which can be a great advantage as the number of labelled examples are often severely restricted. An animal learning about a poisonous food or a predator may have few learning opportunities.

In many cases the complexity of the mapping to be learnt is largely due to the complexity of the input. This is especially true in problems involving perception; it is much easier to learn a mapping from a suitable symbolic representation of 'tiger' to 'run' than to map an array of pixels to the symbolic representation. Unsupervised methods can exploit the statistical regularities of the input by using the large amount of readily available unlabelled examples to learn a mapping from the raw input to a more meaningful internal representation (Barlow 1989).

3 The Hebb unit as suspicious coincidence detector

One of the simplest models of a cell is that of a unit which takes a sum of its inputs (x_j) weighted by the connection strengths (q_j), and gives a positive output (y) when this sum exceeds a given value, its threshold (t):

$$y = 1 \quad \text{if} \quad \Sigma\, q_j x_j > t\,,$$
$$y = 0 \quad \text{otherwise}\,.$$

Such a unit performs a simple kind of pattern matching. If you think of the weights and the inputs as binary patterns then the weighted sum is maximal when the pattern matches the weight vector precisely. Depending on the value of the threshold, the unit will also respond to patterns that differ from the weight vector only in a small number of bits, so this unit can be said to generalize up to a limiting Hamming distance.

This elementary pattern matcher can be made into a suspicious coincidence detector by allowing the connections to change depending on its activity and that of other units to which it is connected. According to a modification rule proposed by Hebb (1949), a connection should become stronger if the two units that it connects are active simultaneously ($\Delta q_j = x_j y$). If on the presentation of a pattern the unit fires, the weights from the active inputs will be strengthened, so the unit will respond to that pattern even better in the future. In this way, the frequently occurring patterns or pattern components are able to tune the weight vector closer to themselves than the infrequent ones. To use several of these units, a mechanism is needed to prevent them from detecting the same feature. One method suggested for the solution of this problem is competitive learning.

4 Competitive learning

Competitive learning (Malsburg 1973; Grossberg 1976) in its simplest version (Rumelhart and Zipser 1985) activates only the unit that fits the input pattern best by selecting the one with the largest weighted sum and suppressing the output of all other units. This can be implemented by strong constant inhibitory connections between the competing units. In this way, the units divide the input space among themselves into disjoint regions, giving a selectively finer discrimination in the regions of space that are densely populated by pattern vectors. The resulting local, 'grandmother-cell' representation can be used by a subsequent supervised layer to associate outputs in a single trial by simply turning on the connections from the winner unit to the active output units. This kind of storage, however, is very limited in the number of discriminable input states that it can code, as well as in its ability to generalize. An output associated to a particular competitive unit gets activated only when the input pattern is within a certain Hamming distance from the weight vector of the unit.

5 Sparse coding

It would be much more desirable to code each input state by a set of active units, each unit representing one component, property or facet of the pattern. Since the combinatorial use of units results in a significant increase in the number of discriminable states, the representational capacity of such a distributed code is high. Distributed representations also give rise to desirable effects like generalisation between overlapping patterns, noise and damage resistance.

On the other hand, when a large number of units are active for each input pattern, the mapping to be implemented by a subsequent layer becomes more complicated and harder to implement by simple neuron-like units. The capacity of an associative memory network, i.e. the number of input-output pairs that can be stored using a highly distributed representation is significantly lower than optimal (Willshaw et al. 1969; Palm 1980). Even more importantly, learning may become extremely slow, and the rules for adjusting connections become complicated and hard to implement (e.g. Rumelhart et al. 1986).

The advantages of both local and distributed representations can be combined by sparse coding, which is a compromise between local and completely distributed representations. In a sparse code, the input patterns are represented by the activity in a small proportion of the available units. By choosing this proportion, one can control the trade-off between representational capacity and memory capacity, as well as that between the amount of generalization and the complexity of the subsequent output function.

As competitive learning is an unsupervised method of forming a local representation, the following mechanism may be considered for coding inputs into a sparse representation.

6 Decorrelation

The mechanism proposed here is one which is aimed at finding a representation in terms of features of components that satisfy the aims stated in Sect. 1. In this model, units within a layer are connected by modifiable inhibitory weights. The development of these feedback weights are governed by an 'anti-Hebbian' modification rule: whenever two units in the layer are active simultaneously, the connection between them becomes more inhibitory, so that joint activity is discouraged in the future and their correlation is decreased (Kohonen 1984; Barlow and Földiák 1989). Training can go on until correlations between the units are completely removed or decreased below a fixed level. In contrast with the 'winner-take-all' mechanism implemented by the strong and fixed inhibitory connections in competitive learning, these modifiable connections allow more than one unit to be active for each pattern, representing it by statistically uncorrelated or not highly correlated set of features.

In a hypothetical problem of coding cars of different colour, the competitive learning scheme would require a separate unit to code each combination of car type and colour (e.g. 'yellow Volkswagen detector' (Harris 1980)), while if car types and colours are not significantly correlated, the above scheme could learn to code colour and type on separate sets of units, and to represent a particular car as a combination of activity in those units (a 'yellow' and a 'Volkswagen' unit). Generalization may then occur specifically along one feature or aspect of the input. An output correlated only with 'Volkswagen' would get connected to the unit in the 'type' group, and it could generalise to other colours even when it has a large Hamming distance from the original.

7 Combination of Hebbian and anti-Hebbian mechanisms

In the following network, the detection of suspicious coincidences is performed by conventional Hebbian feed-forward weights, but units are connected by anti-Hebbian inhibitory feedback connections (Fig. 1). For linear units, this arrangement has been shown to perform principal component analysis by projecting into the subspace of the eigenvectors corresponding to the n largest eigenvalues of the covariance matrix of the input (Földiák 1989).[1] The model discussed here has similar architecture, but units here are nonlinear, so it can learn not only about the second-order statistics, i.e. pairwise correlations between input elements, but also about higher-order dependencies and features of the input.

In order to achieve sparse coding, an additional mechanism is assumed: each unit tries to keep its probability of firing close to a fixed value by adjusting its own threshold. A unit that has been inactive for a long time gradually lowers its threshold (i.e. decreases its selectivity), while a frequently active unit gradually becomes more selective by raising its threshold.

The network has m inputs: $x_j, j = 1 \ldots m$, and n representation units: $y_i, i = 1 \ldots n$. Because of the feedback and the nonlinearity of the units, the output cannot be calculated in a single step as in the case of one unit, because the final output here is influenced by the feedback from the other units. Provided that the feedback is symmetric ($w_{ij} = w_{ji}$), the network is guaranteed to settle into a stable state after an initial transient (Hopfield 1982). This transient was simulated by numerically solving the following differential equation for each input pattern:

$$\frac{dy_i^*}{dt} = f\left(\sum_{j=1}^{m} q_{ij}x_j + \sum_{j=1}^{n} w_{ij}y_j^* - t_i\right) - y_i^*$$

where q_{ij} is the weight of the connection from x_j to y_i, w_{ij} is the connection between units y_i and y_j and the nonlinearity of the units is represented by the function $f(u) = 1/(1 + \exp(-\lambda u))$. The outputs are then calculated by rounding the values of y_i^* in the stable state to 0 or 1 ($y_i = 1$ if $y_i^* > .5$, $y_i = 0$ otherwise). The feedforward weights are initially random,[2] and the feedback weights are 0.

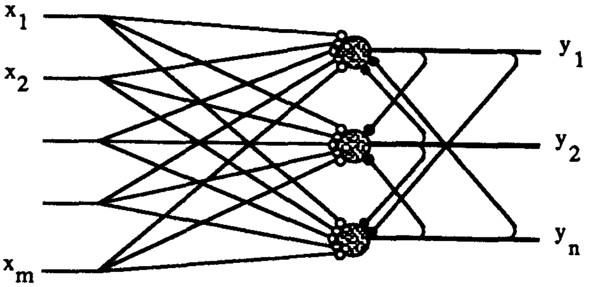

Fig. 1. The architecture of the proposed network. Empty circles are Hebbian excitatory, filled circles are anti-Hebbian inhibitory connections

On each learning trial, after the output has been calculated, the connections and thresholds are modified according to the following rules:

anti-Hebbian rule–

$$\Delta w_{ij} = -\alpha(y_i y_j - p^2)$$

(if $i = j$ or $w_{ij} > 0$ then $w_{ij} := 0$)

Hebbian rule–

$$\Delta q_{ij} = \beta y_i(x_j - q_{ij})$$

threshold modification–

$$\Delta t_i = \gamma(y_i - p)\,.$$

Here α, β and γ are small positive constants and p is the specified bit probability. The Hebbian rule contains a weight decay term in order to keep the feed-forward weight vectors bounded. The anti-Hebbian rule is inherently stable so no such normalizing term is necessary. Note that these rules only contain terms related to the units that the weight connect, so all the information necessary for the modification is available locally at the site of the connection.

In the next two sections some aspects of the model will be demonstrated on two simple, artificially generated distributions.

8 Example 1: learning lines

Patterns consisting of random horizontal and vertical lines were presented to the network. This example was chosen for comparison with that given by Rumelhart and Zipser (1985) to demonstrate competitive learning.

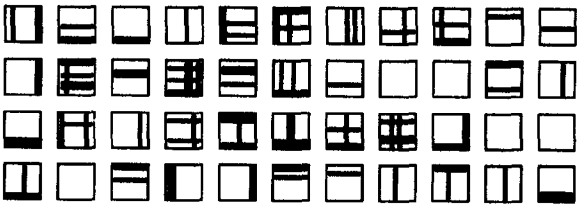

Fig. 2. A random sample of the patterns presented to the network in Example 1

[1] A similar but asymmetrically connected network has also been proposed for this purpose by Rubner and Schulten (1990)

[2] Selected from a uniform distribution on [0, 1] and normalised to unit length ($\Sigma_j q_{ij}^2 = 1$)

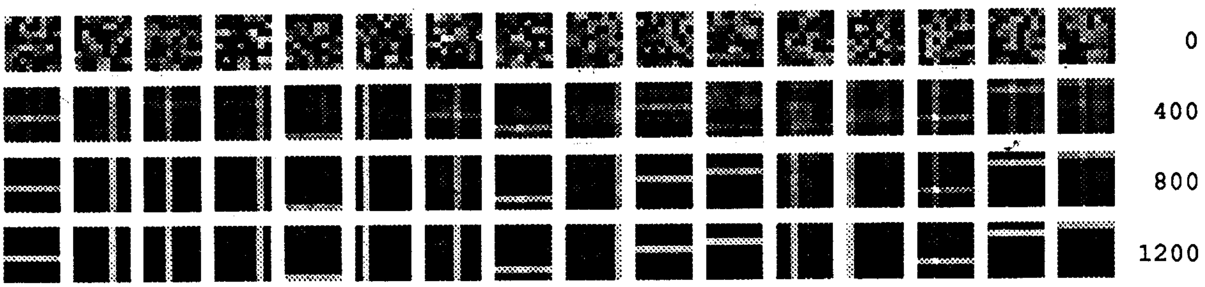

Fig. 3. The feedforward connections of the 16 output units as a function of learning trials in Example 1. $\alpha = 0.1$, $\beta = \gamma = 0.02$, $\lambda = 10$, $p = 1/8$. Thresholds were allowed to reach stable values by running the network with $\alpha = \beta = 0$, $\gamma = 0.1$ for 100 cycles before training

The important difference is that the patterns here consist of combinations of lines. On an 8 × 8 grid, each of the 16 possible lines are drawn with a fixed probability (1/8) independently from all the others (Fig. 2). Pixels that are part of a drawn line have the value 1, all others are 0. The network has 16 representation units.

The feedforward connections developed so that the units became detectors of the most common, highly correlated components, the suspicious coincidences of the set: lines (Fig. 3). Patterns consisting of combinations of lines were coded by a combination of activity in the units. The code generated in this example is optimal in the sense that it preserves all the information in the input, and all the redundancy is removed by the network as the outputs are statistically independent. Of course this is only the case because of the simplicity of the artificial distribution and the fact that the network size was well matched to the number of components (line positions) in the input.

Example 2: learning the alphabet

A slightly more realistic example is considered in this section where the statistical structure of the input is more complicated. This example was chosen for comparison with that presented by Barlow et al. (1989) where methods were considered for uniquely assigning binary strings of a fixed length to a set of probabilities so as to minimise the higher order redundancy of the strings. If A_j is the probability of string j, b_{ij} denotes the ith bit of the code for the jth string and the probability of the ith bit being 1 is p_i, then higher order redundancy can be defined as (Barlow et al. 1989):

$$R = [e(A, b) - E(A)]/E(A),$$

where

$$e(A, b) = -\sum_i [p_i \log p_i + (1 - p_i) \log(1 - p_i)]$$

is the sum of the individual entropies of the bits of the string, and

$$E(A) = -\sum_j A_j \log A_j$$

is the entropy of the set of strings. The sum of the bit entropies is never smaller than the entropy of the

Table 1. The code generated by the network after the presentation of 8000 letters. The rows indicate the output of the 16 units for the input patterns indicated on the right-hand side ($\alpha = 0.01$, $\beta = 0.001$, $\gamma = 0.01$, $\lambda = 10$, $p = 0.1$)

network output	input patterns
0000000000000000	(space)
0000010010000000	e
1000000000000000	t
0000100000000000	i !
0100010000000000	o
0000000010000000	a
0001000000000001	n
0000000000000010	s
0000010000000000	r(– ″ X
0001000000000000	h m
0000110000000000	l T |
0000000000100000	c
0001000000001000	u
0001011000000000	d
1000010000100000	f
0100010000010000	b
0110000000000000	p
0001010010000000	g
0000000100000100	y
0011010000000100	w
0000000000001000	v
0000000000001100	′
0010000000001000	· 1 C G
0001010000000000	N H
0000010000010000	k B R F
0010010000001000	I 4]
0000001101000000	x
0001000010001000	q
0010011000000000	) W
0000011100000000	V
0000001100010000	P
0000011000001000	S 8
0010000000000000	A : <
0000000000000100	;
0010001000001000	2
0000011000011000	O 0 Q U 9
0010010000000000	j / ′ = > %
0010010000010000	E K
0000010000011000	D 6
0011010000000000	M
0010000000011000	L
0010000001000000	z
0000011000000000	3 5
0000010000000100	+ [
0000001000000000	?
0010011100000000	#
0010000100000000	7
0010011000001000	J

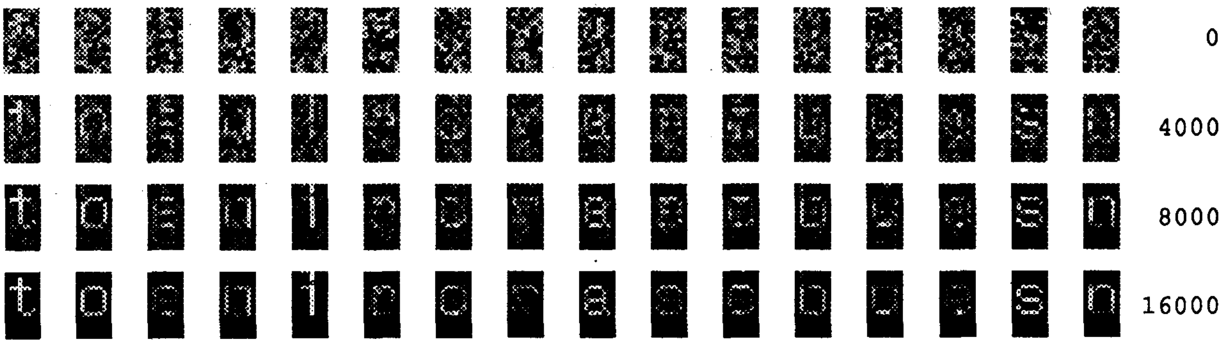

Fig. 4. The receptive fields of the units as a function of the number of letters presented to the network in Example 2. Thresholds were allowed to settle as in Example 1

Table 2. Some properties of the code in Example 2

	input	output
number of units	120 (8 × 15)	16
entropy (E)	4.34 bits	4.22 bits (97% of input)
sum of bit entropies (e)	24.14 bits	5.86 bits
redundancy (R)	456%	39%
bit probabilities	high	low
type of representation	distributed	sparse

strings, and they are equal only when the bits are independent.

The input patterns in this example consist of images of letters presented in a fixed position on an 8 × 15 raster. During training, letters were presented in random order with the same probabilities as they appeared in a piece of English text.[3]

Due to the prescribed bit probability (p), the resulting output patterns contain only a small number of 1's (Table 1). Frequent letters tend to have fewer active bits than infrequent ones, as otherwise the correlations introduced by the frequent simultaneous firing of a large number of cells would force the decorrelating connections to increase inhibition between the active units. Another feature of the code, which is not due to an explicit constraint, is that no two frequent letters are assigned the same output, so that while the code is not completely reversible, it preserved a large proportion (97%) of the information present in the input (Table 2). This is significantly better than the amount of information retained by an untrained random network, which in this example is less than 50%.

A property of the code, which is important from the point of view of generalization, is its smoothness, i.e. that similar input patterns tend to get mapped to similar output patterns (as in the case of letter e and o and even in the confusion of O, 0, Q, U and 9 in Table 1).

The receptive fields of the units reflect the properties of the code. Some of the units detect one of the most frequent letters and become highly selective, while many other units are less selective and their receptive fields consist of different combinations of features in the input patterns (Fig. 4).

10 Discussion

In both examples the network implemented a smooth, information preserving, redundancy reducing transformation of the distributed input patterns into an approximately uncorrelated, sparse activity of units.

What implications does such a code have for generalization in a subsequent supervised layer? It can be observed in both examples that frequent patterns tend to get coded into the activity of a smaller number of units then the infrequent ones. Generalisation therefore works best for infrequent, 'unknown' patterns that are represented as sets of more frequent, 'known' components. For more frequent patterns, the representation tends to be more localized, so output patterns can be associated to them more specifically, without interference from other associations.

Unlike in the case of a linear network, it may be useful to consider a hierarchical arrangement of such subnetworks, each layer extracting different forms of redundancy present in the environment. Such a simple model, of course, does not answer our original question about how a meaningful representation of the world is created in the brain, as it ignores most of the known facts about the genetically determined properties and anatomical constraints of the brain, but it demonstrates one of the possible principles that may underlie the largely unexplained function of the sensory system.

Acknowledgements. I would like to thank Prof. H. B. Barlow for his comments on earlier versions of this paper as well as Dr. G. J. Mitchinson and others in Cambridge for useful discussions. This work was supported by an Overseas Research Studentship, a research studentship from Churchill College, Cambridge and SERC grants GR/E43003 and GR/F34152.

References

Barlow HB (1961) Possible principles underlying the transformations of sensory messages. In: Rosenblith WA (ed) Sensory communication, MIT Press, Cambridge (Mass) London, pp 217–234

[3] Input vectors were constructed from the standard system font of a Sun-3 workstation and vectors were normalized to unit length. The same letter frequencies were used as in Barlow et al. (1989)

Barlow HB (1972) Single units and sensation: a neuron doctrine for perceptual psychology? Perception 1:371–394

Barlow HB (1985) Cerebral cortex as model builder. In: Rose D, Dobson VG (eds) Models of the visual cortex. Wiley, Chichester, pp 37–46

Barlow HB (1989) Unsupervised learning. Neural Comput 1:295–311

Barlow HB, Földiák P (1989) Adaptation and decorrelation in the cortex. In: Durbin RM, Miall C, Mitchison GJ (eds) The computing neuron, chap 4, Addison-Wesley, Wokingham, pp 54–72

Barlow HB, Kaushal TP, Mitchison GJ (1989) Finding minimum entropy codes. Neural Comput 1:412–423

Földiák P (1989) Adaptive network for optimal linear feature extraction. Proceedings of the IEEE/INNS International Joint Conference on Neural Networks, Washington D.C., June 18–22, 1989, vol. 1. IEEE Press, New York, pp 401–405

Grossberg S (1976) Adaptive pattern classification and universal recoding. I. Parallel development and coding of neural feature detectors. Biol Cybern 23:121–134

Harris CS (1980) Insight or out of sight?: Two examples of perceptual plasticity in the human adult. In: Harris CS (ed) Visual coding and adaptability. Erlbaum, Hillsdale, NJ

Hebb DO (1949) The organization of behaviour. Wiley, New York

Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci USA 79:2554–2558

Kohonen T (1984) Self-organization and associative memory. Springer, Berlin Heidelberg New York

Malsburg Ch von der (1973) Self-organization of orientation sensitive cells in the striate cortex. Kybernetik 14:85–100

Palm G (1980) On associative memory. Biol Cybern 36:19–31

Rubner J, Schulten K (1990) Development of feature detectors by self-organization. Biol Cybern 62:13–199

Rumelhart DE, Zipser D (1985) Feature discovery by competitive learning. Cogn Sci 9:75–112

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. In: Rumelhart DE, McClelland J (eds) Parallel distributed processing, vol 1. MIT Press, Cambridge Mass London, pp 318–362

Watanabe S (1960) Information-theoretical aspects of inductive and deductive inference. IBM J Res Dev 4:208–231

Watanabe S (1985) Pattern recognition: human and mechanical. Wiley, New York

Willshaw DJ, Buneman OP, Longuet-Higgins HC (1969) Non-holographic associative memory. Nature 222:960–962