5 Adaptive Bidirectional Associative Memory (ABAM)

learning objectives

- the differences between two types of learning methods: supervised and unsupervised
- how signals (training data) may be sent forward **and** backward through a network
- how a network "learns" by changing the weights of its neurons
- how matrices are used to represent and modify weights
- how a trained network can actually recognize separated parts of a stored pattern

5.1 Unsupervised and Supervised Learning

When learning is *unsupervised*, the system is provided with a group of facts (patterns) and then left to itself to settle down (or not!) to a stable state in some number of iterations (Figure 5-1). Inherent in any unsupervised learning system is an optimization (or decision) criterion that is used for the evaluation of the result at the end of each cycle. This, however, is a very general one, such as minimization of energy or distance, maximization of profit, etc.

Thus, learning is basically an optimization procedure. An example of this kind of learning is *hierarchical clustering*, in which we have a set of objects or patterns $\{X_s\}$ and two criteria, the distance between two objects and the distance between two groups of objects. We want a system which will, in response **only** to these conditions, organize the objects into groups and the groups into a hierarchy. The result should be a surprise, or at least not be influenced by our expectations.

Suppose, however, that we have some objects (patterns, say) whose behavior (responses) in a given system are known. The two types of data (the representation of the objects and their responses in

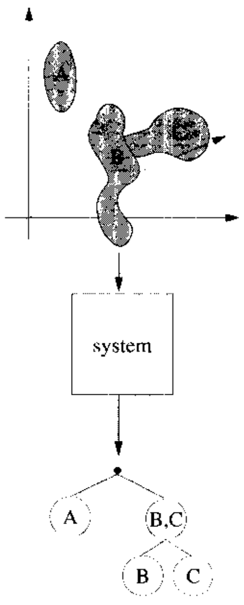

Figure 5-1: Unsupervised learning.

the system) form pairs of what we might call inputs and targets. The goal of *supervised methods* is to find a *model* – a general procedure – that will correctly associate the inputs with the targets (Figure 5-2).

In a sense, the targets do not enter the learning procedure; they only serve as a criterion for how well the system has been trained. According to this estimate, the decision is made whether the network (specifically, the weights) need to be corrected (see Section 2.3). Of course, the change or correction of each weight depends on the size of the error produced, which in turn depends on the target; but mainly the correction is proportional to the input.

At the moment we are not concerned with how to change the weights; let's concentrate on the idea that the weights are forced to change to give a specific answer defined by the user.

When applying any supervised learning method, we have to distinguish two cases that differ from each other in the way the target is related to the input; specifically, whether some intrinsic relationship occurs between them, or they are just arbitrarily associated. Examples of arbitrary associations are the number of eyes most of us have, and the symbol for that number ("10" in the binary system, "2" in others, such as the decimal system); or the sound you make when you smile and force air between your teeth, and the letter "c" (cf. Figure 5-3).

On the other hand, the pair "chemical structure" and its "infrared spectrum" have an intrinsic relationship, because the structure of the molecule causes the spectrum whether or not we know the mechanism.

Now, when considering both types of pairs, the unrelated and related ones, it is evident that supervised learning will be used for different purposes with different types of pairs. In the case of unrelated pairs, it will mainly be used for identifying corrupted patterns, while in the case of related pairs it will be used for building models that can predict responses for different patterns from those used in the learning process.

When the pairs are unrelated, it is obviously not possible to generalize the solution: it is hard to imagine that after learning to associate the images of a cat, a dog and a fish with the letters "C", "D" and "F", the system will produce an "H" for the image of a horse.

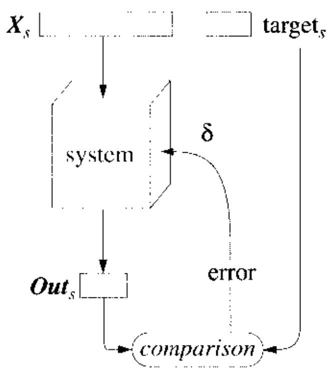

Figure 5-2: Supervised learning.

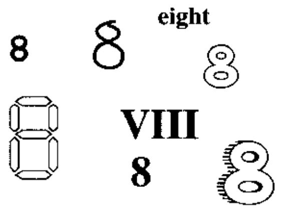

Figure 5-3: Arbitrary relations between objects: a number, and the symbols used for it.

But this conclusion requires a word of caution. At one level, "cow" and "milk" are unrelated – they don't even have any letters in common – but at another level, they are. That is, there might be no generalization on the level at which the system was trained, while one does exist on some higher (or, if you prefer, lower) level for which the system was not intended to make predictions. If this turns out to be true the ABAM net is worth considering. Below, we will give some examples to demonstrate the point.

5.2 General

The word "associative" in the name *adaptive bidirectional associative memory* (ABAM) indicates that it is a system that can learn to associate patterns. In a way, ABAM resembles the Hopfield network: both are one-layer neural networks, and both were developed to manipulate binary (or bipolar) vectors and have therefore an identical evaluation of the initial weights. After this, however, the similarity breaks down.

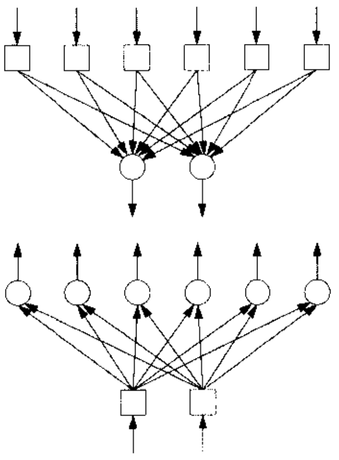
Figure 5-4: The two directions of signal propagation in an ABAM network (cf. also Section 5.4).

The most important difference between the two networks lies not in their architecture, but in the scope of the problems they can tackle and how the networks (or weights) are adapted to these problems. The entire Hopfield network (the number of neurons, and the number of weights and their values), is determined once and for all after the patterns or objects to be learned have been chosen. The size of the image, i.e. the number of pixels in each pattern, determines the number of neurons and the number of weights (see Section 4.2), while the number and the form of the selected patterns determine the values of all the weights (see Equation (4.5)).

This does not work with the ABAM network. First, it is not necessarily square, but can be rectangular if convenient. This is because there can be fewer neurons in the output layer than there are in the non-active input layer (Figure 5-4). This can save a lot of computer memory, as well as computation time.

Second, although the initial setup of the weights is made in the same way as for the Hopfield net (compare Equations (5.2) and (4.5)), in the ABAM network they are used only in the first iteration step. After this, they are changed (adapted).

> This is what the "adaptive" in the name ABAM stands for.

How and why are the weights changed?

Let us turn to the second question first. The weights in the ABAM network are changed in order to give an exact and **predetermined** response that is not the same as the input pattern. This means that the ABAM network can learn to associate **unrelated pairs** of patterns. Thus, the ABAM network is able to associate patterns with their written or spoken descriptions, and, as in the case of the Hopfield network, the trained ABAM network will still be able to reproduce the associated pattern even from corrupted input. Such pairs of patterns, the inputs and the targets, are essential for supervised learning.

5.3 ABAM Network

The ABAM network is a one-layer network. The number of neurons in the active layer is often considerably smaller than the number of inputs (Figure 5-5).

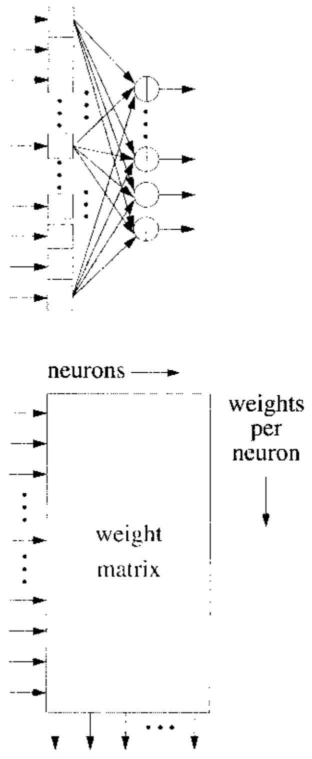

Figure 5-5: The architecture of the ABAM network: it is very similar to the Hopfield network, but usually has many fewer neurons on the output side than on the input.

The first step, of course, is to generate the input vectors X_s with their corresponding outputs Y_s (targets). Let there be p pairs, each consisting of one m-dimensional input and one n-dimensional target vector. Initially, we will consider these vectors to be bipolar (components, equal to ±1). The pairs (X_s, Y_s) can be written:

$$X_s = (x_{s1}, x_{s2}, \ldots, x_{si}, \ldots, x_{sm})$$

and (5.1)

$$Y_s = (y_{s1}, y_{s2}, \ldots, y_{sj}, \ldots, y_{sn})$$

The initial weights, $W^{(0)}$, are calculated from the set of p pairs of input and target vectors, $\{X, Y\}$:

$$w_{ji}^{(0)} = \sum_{s=1}^{p} x_{si} \cdot y_{sj} \tag{5.2}$$

The symbol { } denotes a group of objects, in our case a group of pairs consisting of input and target vectors X and Y. Be sure to distinguish this notation from that for a single pair of vectors (X_s, Y_s).

5.4 Learning Procedure

The basis of learning in the ABAM network is the fact that an (m x n) matrix can be multiplied from two directions: in the standard way by an m-dimensional vector, or in transposed form (reflected across the main diagonal) by an n-dimensional vector. In the language of neural networks, the side of the weight matrix W having the same number of rows (or columns) as the input vector is called the *input* side, and the other, the *output* side.

This leads to the very nice property that an "input" object X can produce the "output" $Y^{(1)}$; or a "target" Y if inputted on the **output** side can produce an "output" vector $X^{(1)}$ on the, formally speaking, **input** side of the weight matrix. (The quote around "input", etc., reflect the fact that input and output are not fixed with respect to the matrix any more. See Figure 5-6)

Thus, **any** pair of objects (X, Y) of different dimensions may be paired up with another pair having the same dimensions $(X^{(1)}, Y^{(1)})$. Because both multiplications can be regarded as sending signals (X or Y) through the neural network in opposite directions, it is called *bidirectional*.

Combining the procedure that generates the weight matrix W from a **set** of pairs of objects $\{X, Y\}$ (Equation (5.2)) with the bidirectional procedure for generating pairs of vectors, we obtain an iterative learning scheme:

The iterative procedure (5.4) stops when the weight matrix $W^{(t)}$ produces a set of pairs $\{X^{(t)}, Y^{(t)}\}$ identical to the initial set $\{X, Y\}$.

It must be emphasized that there is **no guarantee** that such a weight matrix can be found for an arbitrary set of pairs. As with the Hopfield network, extreme care should be used when selecting the appropriate representations of input patterns and targets.

We will now describe the iterative procedure in detail. Each iteration step t is composed of three parts:

- the generation of a new weight matrix $W^{(t)}$ from the set of pairs $\{X^{(t)}, Y^{(t)}\}$

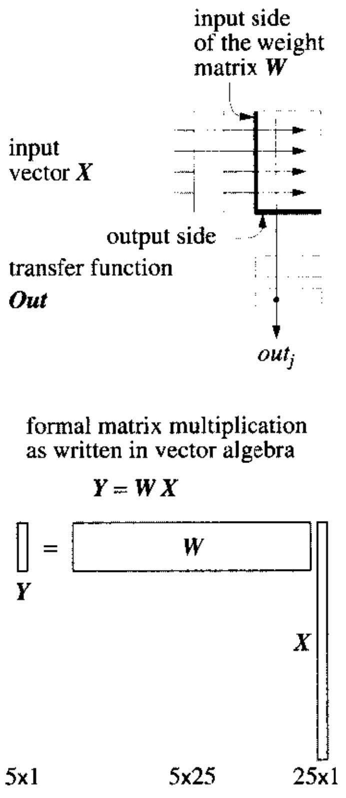

Figure 5-6:
a) Multiplication of a weight matrix by the input vector X in order to obtain the output vector Y.
(b) Multiplication of the transposed matrix W^T by the target vector Y in order to obtain a vector X on the "input" side of the matrix W.

If we want to cut down the dimensions of the neural network, the dimensions of the targets should be smaller than those of the input vectors. For this example we have decided to use five-element vectors as targets. Figure 5-8 shows the inputs (X_1, X_2, X_3, X_4 and X_5) and their corresponding target patterns (Y_1, Y_2, Y_3, Y_4 and Y_5).

As usual, each input image is coded as a 25-element vector, beginning at the upper left and proceeding row by row to the lower right. Each target is coded as a five-element vector of 1's and 0's; each position in this vector identifies a different target. (Before entering the ABAM network, these are converted into bipolar coding.)

Equation (5.2) produces the (25 x 5)-element matrix shown in Table 5-1. Fortunately, in this case the learning procedure (Equation (5.4)) required no iteration, which means that the matrix $\boldsymbol{W}$ was not modified.

In applying the network, the output layer can be regarded as a switch with five positions, depending on which pattern was input.

Okay, but so what? Many other conventional methods can do this just as well. What is interesting about this mechanism is that it recognizes the input patterns (i.e., answers with the correct output signals) even if **corrupted** patterns are input.

Changing one pixel out of 25 represents a 4% error. Now, this error can occur at 25 different places; if we check the responses to all possible 1-pixel errors on each of the 5 images, we get the very interesting results given in Table 5-2.

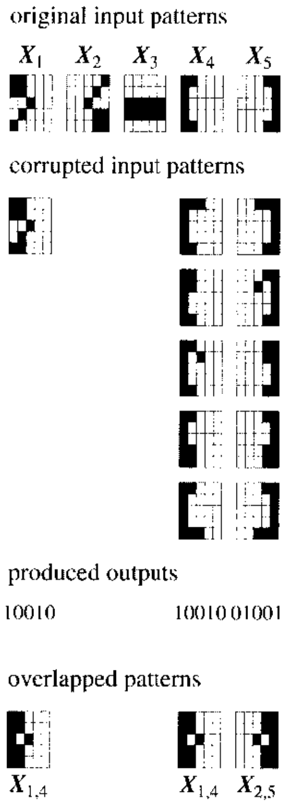

Figure 5-9: The eleven corrupted patterns (from among all 125 one-pixel errors) that caused the firing of two neurons instead of one.

pattern	target	actual output			
1	10000	10000	(24 times)	10010	(once)
2	01000	01000	(25 times)		
3	00100	00100	(25 times)		
4	00010	00010	(20 times)	10010	(5 times)
5	00001	00001	(20 times)	01001	(5 times)

Table 5-2: Responses to all possible 1-pixel errors in the images shown in Figure 5-8.

Although in all 125 cases the correct bit is set in the output, an additional bit is set in eleven of them. So, we are led to ask if there is anything special about the eleven corrupted patterns that caused this additional output neuron to fire. These are shown in Figure 5-9: one has an error in X_1, five in X_4 and five in X_5.

(In the following, we will use the symbol $X_{i,j}$:

$$X_i \text{ OR } X_j = X_{i,j} \qquad (5.6)$$

to designate the overlap, or logical OR, of the patterns X_i and X_j.)

Look at the corrupted X_1, which activates the first and fourth bits in the target; it is very similar to $X_{1,4}$, which is the overlap of the two patterns X_1 and X_4 shown at the bottom of Figure 5-9; the difference between $X_{1,4}$ and the corrupted X_1 is only two pixels.

Interestingly, all five corrupted X_4's activate these same two bits in the output; the difference between $X_{1,4}$ and the five corrupted X_4's is either two or four pixels. And finally the five corrupted X_5's shown in the rightmost column of Figure 5-9 all activate the second and fifth bits in the output. The corresponding pattern overlap is shown at the bottom right of Figure 5-9; note its similarity to the five corrupted inputs. The reader can verify the outputs of the images corrupted by the 1-pixel errors by the use of program ABAM as described in Paragraph 5.6.

So, even when the ABAM network makes errors, they are reasonable ones. Encouraged by this, we can set up an additional experiment to show the power of the method. Ten different overlapping pairs (Figure 5-10) can be made from five different input patterns; let us construct the bipolar 25-element representations of these overlapped input patterns, input them to the matrix $\boldsymbol{W}$ and watch the 5-element output.

The combination $X_{1,2}$ is shown explicitly below:

$$X^{binary}_{1,2} = \begin{matrix}(1, 1, 0, 0, 1,\\ 1, 1, 0, 1, 0,\\ 0, 0, 1, 0, 0,\\ 0, 1, 0, 1, 1,\\ 1, 0, 0, 1, 1)\end{matrix} \qquad X^{bipolar}_{1,2} = \begin{matrix}(\ 1, & 1, & -1, & -1, & 1,\\ 1, & 1, & -1, & 1, & -1,\\ -1, & -1, & 1, & -1, & -1,\\ -1, & 1, & -1, & 1, & 1,\\ 1, & -1, & -1, & 1, & 1)\end{matrix} \qquad (5.7)$$

In all ten cases the resulting output is exactly what we expected: each input pattern triggered the two neurons associated with its component images. In only one of the ten cases was a third, unrelated neuron fired; this is the combination of the third and fifth object, i.e., $X_{3,5}$. The additional bit which was triggered is bit No. 2. A three-bit output corresponds to the overlap (logical OR) of three of the basic

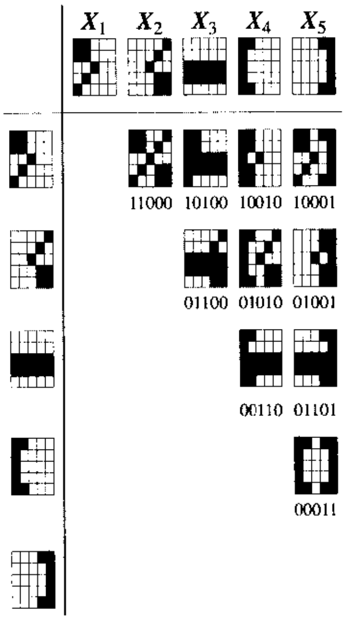

Figure 5-10: The ten possible combinations of two images from five different patterns. The corresponding output vector is shown below each overlapped pattern.

inputs; Figure 5-11 compares these with the actual input pattern that triggers them; the difference is only one pixel.

This example, which can be exercised by the use of program ABAM as described in Paragraph 5.6, shows four capabilities of the ABAM neural network:

- it associates the five input patterns with the five targets, even for corrupted inputs;
- it produces the original input vectors from the corresponding outputs;
- it recognizes the components of two ORed patterns; and
- it produces the negative (complemented) target when negative patterns are input (Figure 5-12).

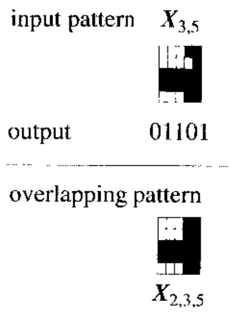

Figure 5-11: The comparison of the two-image input pattern $\mathbf{X}_{3,5}$ that fires three bits, 01101, with the corresponding three-image pattern.

By analogy with the biological brain, these can be interpreted as:

- the ability to recognize learned images by activating the appropriate neuron even if the trigger image is distorted or incomplete;
- the ability to associate a complex pattern with a single neuron (which is fired when that pattern is recognized);
- the ability to identify the parts of a complex pattern, and
- understanding the concept of a "negative" image.

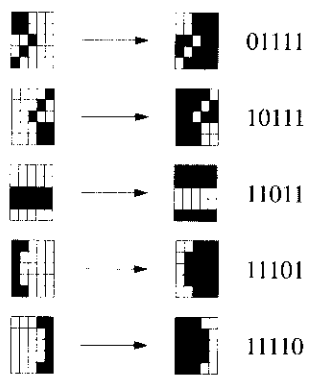

Figure 5-12: The set of five patterns (left) and their negatives (right). Negative (complemented) inputs produce the complements of the corresponding outputs.

5.6 Significance of the Example

Besides these demonstrated features of the simple ABAM network, there is an even more significant lesson to be drawn from this experiment. We have seen, on a small scale, what we hope to obtain from neural networks on a large scale. The effect we are talking about can be linked to the warning (given in the last paragraph of Section 5.1) that "hidden" relationships may exist among seemingly unrelated patterns.

We have taught the system to associate several pairs of obviously unrelated patterns, and nothing more. But the system has learned two **general** concepts that we never taught it: analyzing a sum into its parts, and complementing a result to match a complemented input.

The result of this experiment and the generalizations we have drawn from it, result from the (very careful) choice of patterns; there is little chance that an arbitrary selection of patterns will produce an ABAM net intelligent enough to resolve two ORed patterns. However, it is worth mentioning that ABAM learning is at least sometimes capable of generalization.

In spite of all such precautions, we must admit that this small neural network (an "artificial brain", if you like) consisting of only five neurons with 25 weights each (125 synapses altogether) has done a remarkably good job.

There are, of course, many questions about the performance of such nets when applied to larger problems (more patterns or more pairs of patterns), and we must **always** keep in mind that a net's response is always a result of the entire assembly of patterns. A neural network is in this sense, **holographic**.

Intuition tells us that similar patterns should give similar responses. In most cases this will be true. However, as mentioned before, the way we think about "similarity" between the samples does not necessarily correspond to the similarity found by the network, leading to unexpected results.

Let us go back to Figure 5-9. We might expect that if a given error in pattern X_1 causes the firing of the two neurons 1 and 4, an analogous error in X_2 should also cause the firing of two neurons. Figure 5-13 shows the error in X_2 analogous to the error in X_1. However, this corrupted signal did not produce the two signal output of firing neurons 2 and 5, as might have been expected. Why didn't this happen?

The reason for this is that we've confused "similarity" (a vacuous, undefined term) with "symmetry".

The patterns X_1 and X_2, are related by symmetry (specifically, an inversion center, denoted C_i; that is, taking every point in X_i and moving it to the opposite side of the center point: produces X_2); these are different figures, and can not be superimposed. X_4 and X_5 are related in the same way. The OR of X_1 and X_2, when subjected to this operation, produces a figure which is congruent to itself. Thus, we say that the ORed figure contains an inversion center, while the individual figures, X_1, and X_2, do not.

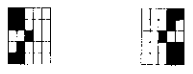

Figure 5-13: The images X_1 and X_2 each with a one-pixel error.

However, the symmetry relations between input and output patterns are products of the entire ABAM matrix, which, in turn, results from all input patterns. In our example, since X_1 and X_2 together possess C_i, they generate a C_i in the network (i.e., they make its **responses** symmetrical); X_4 and X_5 do the same. But X_3 is unique in that it neither possesses the C_i symmetry, nor can combine with another member of the set to produce it; hence, X_3 prevents the net from having this symmetry.

This is also why the overlapping patterns $X_{1,3}$ and $X_{2,3}$ bear no similarity at all in spite of the fact that they were produced by overlapping the object X_3 with two very "similar" (actually "symmetrical") objects X_1 and X_2. On the other hand, the patterns $X_{1,4}$ and $X_{2,5}$ appear similar to the eye because they are related by an inversion center (or plane of symmetry).

Thus, the source of similar effects with corrupted patterns depends on the actual symmetry of the original patterns and on the symmetry of the composites (Figure 5-14).

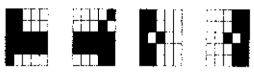

Figure 5-14: The overlapping patterns $X_{1,3}$ and $X_{2,3}$ are not symmetric; $X_{1,4}$ and $X_{2,5}$ are symmetrically related.

In summary, our expectation that X_2 would behave like X_1 was based on their symmetrical relationship; we did not take into account the lack of symmetry (due to X_3's contribution to the weights) of the network itself.

Never trust your impressions about the relation between the objects and their apparent symmetry; find someone who understands symmetry (spectroscopist or a crystallographer is your best bet).

5.7 Essentials

- hetero-association is the main goal of the ABAM network
- the number of output data is usually smaller than the number of input data
- learning, i.e. evaluation of weights, is supervised and achieved by bidirectional iteration, in which the input signals flow towards the output side, forming output data, while the output data are returned in the opposite direction towards the input side forming new inputs
- corrupted input patterns can retrieve the corrected associated patterns from the network
- in certain circumstances the ABAM network can generalize what it has learned

- **weights**

$$w_{ji}^{(0)} = \sum_{s=1}^{p} x_{si} y_{sj} \tag{5.2}$$

- **iterative learning**

start:

$$\begin{tikzcd}
\{X, Y\} \arrow[r] & \boldsymbol{W}^{(0)} & & & \\
& XW^{(0)} \arrow[r] & Y^{(1)} & & \\
& YW^{(0)\mathrm{T}} \arrow[r] & X^{(1)} & & \\
& & \{X^{(1)}, Y^{(1)}\} \arrow[r] & W^{(1)} & \\
& & & XW^{(1)} \arrow[r] & Y^{(2)} \\
& & & YW^{(1)\mathrm{T}} \arrow[r] & X^{(2)} \\
& & & \vdots & \vdots \\
& & & XW^{(t)} \arrow[r] & Y^{(t+1)} \\
& & & YW^{(t)\mathrm{T}} \arrow[r] & X^{(t+1)}
\end{tikzcd}$$

so that $\{X^{(t)}, Y^{(t)}\} \equiv \{X, Y\}$ (5.4)

5.8 References and Suggested Readings

5-1. B. Kosko, "Adaptive Bidirectional Associative Memories", *Appl. Optics* **26** (1987) 4947 – 4960.

5-2. B. Kosko, "Constructing an Associative Memory", *Byte*, September 1987, 137 – 144.

5-3. B. Kosko, "Bidirectional Associative Memories", *IEEE Trans. Syst., Man and Cyber.* **18** (1988) 49 – 60.

5-4. B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1992.

5-5. M. Otto and U. Hörchner, "Application of Fuzzy Neural Networks to Spectrum Identification", in *Software Development in Chemistry 4*, Ed.: J. Gasteiger, Springer-Verlag, Berlin, FRG, 1990, pp. 377 – 384.