POLYTECHNIC UNIVERSITY
Department of Computer and Information Science

Pattern Association

K. Ming Leung

Abstract: Neural networks for pattern association and associative memory are discussed. Also discussed is an important attribute of associative networks called the capacity.

Directory

Last Revision Date: March 10, 2009

Table of Contents

1. Introduction

To a certain extend, learning always involves the forming of associations between stimuli and their corresponding responses. Input stimuli which are similar to the stimulus for the association will invoke the associated response pattern. We will consider some relatively simple single-layer NNs that can learn a set of pattern pairs (or associations): $\mathbf{s}^{(q)} : \mathbf{t}^{(q)}, q = 1, 2, \ldots, Q$. In general, the training vectors $\mathbf{s}^{(q)}$ have N components,

$$\mathbf{s}^{(q)} = \left[\begin{array}{cccc} s_1^{(q)} & s_2^{(q)} & \ldots & s_N^{(q)} \end{array} \right],$$

and their targets $\mathbf{t}^{(q)}$ have M components,

$$\mathbf{t}^{(q)} = \left[\begin{array}{cccc} t_1^{(q)} & t_2^{(q)} & \ldots & t_M^{(q)} \end{array} \right].$$

Unlike classification problems, we cannot expect that $N \gg M$ here for pattern association. We use bipolar neurons so that the components of $\mathbf{s}^{(q)}$ and $\mathbf{t}^{(q)}$ have values of ± 1 only.

We assume here that our NNs have an input layer, an output layer, and no hidden layer. The transfer functions are assumed to be given

by

$$f_\theta(x) = \begin{cases} +1, & \text{if } x > \theta, \\ 0, & \text{if } -\theta \leq x \leq \theta, \\ -1, & \text{if } x < -\theta. \end{cases}$$

where $\theta = 0$.

There are two types of associations:

1. The case where $\mathbf{s}^{(q)} = \mathbf{t}^{(q)}, q = 1, 2, \ldots, Q$, are referred to as an auto-associative memory.
2. otherwise it is referred to as hetero-associative memory.

There are two types of architectures:

1. feedforward: signals go from the input layer to the output layer in one direction only.
2. recurrent: closed loops exist within the NN enable signals to circulate among neurons or even within single neurons.

Some remarks:

1. An associative memory net may serve as a simplified model of human memory.
2. Associative memory also provides an approach to storing and retrieving data based on content rather than storage address (*i.e.* content addressable memory).
3. Information stored are distributed throughout the NN (in the weights and biases).
4. Associative memory is naturally fault-tolerant without explicit redundancy.

2. Hebb rule Training

We assume that the initial weights and biases are zero, and that the biases are absorbed by introducing an extra neuron, X_0. We assume also that the NN is trained using Hebb's rule, and so the weight matrix

(which may contain the biases as well) after training is

$$\mathbf{W} = \sum_{q=1}^{Q} \mathbf{s}^{(q)T}\mathbf{t}^{(q)} = \sum_{q=1}^{Q} \begin{bmatrix} s_1^{(q)} \\ s_2^{(q)} \\ \dots \\ s_N^{(q)} \end{bmatrix} \begin{bmatrix} t_1^{(q)} & t_2^{(q)} & \dots & t_M^{(q)} \end{bmatrix}.$$

Now we check to see whether this NN can correctly produce the output pattern if one of the training vector, say $\mathbf{x} = \mathbf{s}^{(k)}$ is presented to it. Using matrix notation, we have

$$\mathbf{y}_{\text{in}} = \mathbf{s}^{(k)} \sum_{q=1}^{Q} \mathbf{s}^{(q)T}\mathbf{t}^{(q)} = \mathbf{s}^{(k)}\mathbf{s}^{(k)T}\mathbf{t}^{(k)} + \sum_{q\neq k}^{Q} \left(\mathbf{s}^{(k)}\mathbf{s}^{(q)T}\right)\mathbf{t}^{(q)}.$$

If the training vectors are uncorrelated (that means that they are mutually orthogonal), then the dot-product

$$\mathbf{s}^{(k)}\mathbf{s}^{(q)T} = 0, \quad \text{if } q \neq k.$$

In that case,

$$\mathbf{y}_{\text{in}} = \|s^{(k)}\|^2\mathbf{t}^{(k)}.$$

Since $\|s^{(k)}\|^2$ is positive, we have

$$\mathbf{y} = f(\mathbf{y}_{\text{in}}) = t^{(k)},$$

and this is the desired output. Thus if the training vectors are uncorrelated then the NN can always make the correct associations.

If the training vectors are not uncorrelated, but if the correlations are small enough that the dot-products between training vectors are not large enough to alter the sign of the first term in $\mathbf{y}_{\text{in}}$, then the network may still produce the correct outputs. Notice that in the case of bipolar neurons, $\|s^{(k)}\|^2 = N$, which is typically a large positive number. If the training vectors are not too highly correlated, then the dot-products are relatively small. They are expected to have different signs and therefore partially cancellation is expected among the terms in the above sum.

2.1. An Example for Heteroassociative Memory

We now consider using Hebb's rule for the following training set.

q	$\mathbf{s}^{(q)}$	$t^{(q)}$
1	$\begin{bmatrix} 1 & -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & -1 \end{bmatrix}$
2	$\begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & -1 \end{bmatrix}$
3	$\begin{bmatrix} -1 & -1 & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 1 \end{bmatrix}$
4	$\begin{bmatrix} -1 & -1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 1 \end{bmatrix}$

From the set of targets, it is clear that the two biases are zero.

Hebb's rule gives the weight matrix

$$\begin{aligned}
\mathbf{W} &= \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} \\
&\quad + \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 4 & -4 \\ 2 & -2 \\ -2 & 2 \\ -4 & 4 \end{bmatrix}
\end{aligned}$$

We check to see if the correct output will be produced when the

training vectors are presented to the NN. We find that for $q = 1$

$$\mathbf{y}_{\text{in}} = \begin{bmatrix} 1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 4 & -4 \\ 2 & -2 \\ -2 & 2 \\ -4 & 4 \end{bmatrix} = \begin{bmatrix} 8 & -8 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & -1 \end{bmatrix},$$

for $q = 2$

$$\mathbf{y}_{\text{in}} = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 4 & -4 \\ 2 & -2 \\ -2 & 2 \\ -4 & 4 \end{bmatrix} = \begin{bmatrix} 12 & -12 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & -1 \end{bmatrix},$$

for $q = 3$

$$\mathbf{y}_{\text{in}} = \begin{bmatrix} -1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 4 & -4 \\ 2 & -2 \\ -2 & 2 \\ -4 & 4 \end{bmatrix} = \begin{bmatrix} -8 & 8 \end{bmatrix} \Rightarrow \begin{bmatrix} -1 & 1 \end{bmatrix},$$

and for $q = 4$

$$\mathbf{y}_{\text{in}} = \begin{bmatrix} -1 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 4 & -4 \\ 2 & -2 \\ -2 & 2 \\ -4 & 4 \end{bmatrix} = \begin{bmatrix} -12 & 12 \end{bmatrix} \Rightarrow \begin{bmatrix} -1 & 1 \end{bmatrix}.$$

Thus the NN makes the correct associations for each of the vectors in the training set.

Note that the matrix, whose ij element is the dot-product between the training vectors, $\mathbf{s}^{(i)}$ and $\mathbf{s}^{(j)}$, is given by

$$\begin{bmatrix} 4 & 2 & 0 & -2 \\ 2 & 4 & -2 & -4 \\ 0 & -2 & 4 & 2 \\ -2 & -4 & 2 & 4 \end{bmatrix}$$

The diagonal elements 4 are the square of the magnitudes of the training vectors. Only the first and the third training vectors are orthogonal to each other. Thus there are cross talks, but their presence are not significant enough to destroy the correct association for the

original training set.

Next we consider input vector that the NN has not seen before. For example, we take the second training vector $\mathbf{s}^{(2)} = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}$ and flip its third element to obtain $\begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}$. Presenting this vector to the NN gives $\mathbf{y}_{\text{in}} = \begin{bmatrix} 8 & -8 \end{bmatrix}$, and therefore the output is $\mathbf{y} = \begin{bmatrix} 1 & -1 \end{bmatrix}$. This is the correct association for training vector 2.

Suppose we zeroth out the first and the third element of $\mathbf{s}^{(2)}$ (to indicate uncertainties in those values), to obtain an input vector $\begin{bmatrix} 0 & 1 & 0 & -1 \end{bmatrix}$, then we find $\mathbf{y}_{\text{in}} = \begin{bmatrix} 6 & -6 \end{bmatrix}$. This gives the correct associated output $\mathbf{y} = \begin{bmatrix} 1 & -1 \end{bmatrix}$. Notice that this is the most reasonable association that can be make (by a NN or by a human).

However, if we zeroth out the first and the fourth element of $\mathbf{s}^{(2)}$, to obtain an input vector $\begin{bmatrix} 0 & 1 & -1 & 0 \end{bmatrix}$, then we find $\mathbf{y}_{\text{in}} = \begin{bmatrix} 0 & 0 \end{bmatrix}$, which gives a totally uncertain output $\mathbf{y} = \begin{bmatrix} 0 & 0 \end{bmatrix}$. For this input vector, the NN or any human has no way of knowing what the correct associated output should be. It resembles some of the training vectors whose associated output is $\mathbf{y} = \begin{bmatrix} 1 & -1 \end{bmatrix}$, as well as some of the training vectors whose associated output is $\mathbf{y} = \begin{bmatrix} -1 & 1 \end{bmatrix}$.

3. Extended Delta Rule for Multiple Output Neurons

The Delta rule can also be used to train a NN for pattern association. We now extend the previous Delta rule for single layer NN with multiple output neurons to the case where the transfer function during training is not an identity function. Thus the output by neuron $\mathbf{Y}_j$ is

$$y_j = f(y_{\text{in,j}}) = f(\sum_{i=0}^{N} s_i(k) w_{ij}(k)),$$

where f may no longer be the identity function. The derivation of the Delta rule involves the taking of partial derivatives. Thus during training at least, we cannot use Heaviside step functions as transfer functions, we need to use some other functions that possess derivatives.

We will absorb the biases as we did before. Suppose at the k-th step in the training process, the current weight matrix and bias vector are given by $W(k)$ and $\mathbf{b}(k)$, respectively, and one of the training vectors $\mathbf{s}(k) = \mathbf{s}^{(q)}$, for some integer q between 1 and Q, is presented to the NN. Since the target is $t_j(k) = t_j^{(q)}$, and so the error is $y_j - t_j(k)$

for the j-th output component. Thus for the Delta rule, we want to find a set of w_{mn} that minimizes the quantity

$$E(\mathbf{W}(k)) = \sum_{j=1}^{M} (y_j(k) - t_j(k))^2 = \left(f(\sum_{i=0}^{N} s_i(k) w_{ij}(k)) - t_j(k) \right)^2 .$$

We take the gradient of this function with respect to w_{mn}

$$\begin{aligned} \partial_{w_{mn}} E(\mathbf{W}(k)) &= \partial_{w_{mn}} \sum_{j=1}^{M} (y_j(k) - t_j(k))^2 \\ &= 2 \sum_{j=1}^{M} (y_j(k) - t_j(k)) \, \partial_{w_{mn}} y_j(k). \end{aligned}$$

Using the chain rule for differentiation, we have

$$\partial_{w_{mn}} y_j(k) = \partial_{w_{mn}} f(\sum_{i=0}^{N} s_i(k) w_{ij}(k)) = f'(y_{\text{in,j}}) \sum_{i=0}^{N} s_i(k) \partial_{w_{mn}} w_{ij}(k)$$

Since

$$\partial_{w_{mn}} w_{ij}(k) = \delta_{i,m} \delta_{j,n},$$

thus

$$\partial_{w_{mn}} y_j(k) = f'(y_{\text{in,j}}) \sum_{i=0}^{N} s_i(k)\delta_{i,m}\delta_{j,n} = \delta_{j,n} f'(y_{\text{in,j}}) s_m(k)$$

we have

$$\begin{aligned} \partial_{w_{mn}} E(\mathbf{W}(k)) &= 2 \sum_{j=1}^{M} \left(y_j(k) - t_j(k)\right) \delta_{j,n} f'(y_{\text{in,j}}) s_m(k) \\ &= 2 s_m(k) \left(y_n(k) - t_n(k)\right) f'(y_{\text{in,j}}). \end{aligned}$$

Therefore if the steepest descent method is used to adjust the weights and biases, then we have

$$\mathbf{w}_{ij}(k+1) = \mathbf{w}_{ij}(k) - 2\alpha s_i(k) \left(y_j(k) - t_j(k)\right) f'(y_{\text{in,j}}).$$

The $i = 1, 2, \ldots, N$ components of this equation gives the updating rule for the weights. The $i = 0$ component of this equation gives the updating rule for the biases

$$b_j(k+1) = b_j(k) - 2\alpha \left(y_j(k) - t_j(k)\right) f'(y_{\text{in,j}}).$$

If the transfer function is the identity function, then $f' = 1$, these equations reduce to those we obtained before.

4. Autoassociative Net

For autoassociative nets, we have $\mathbf{s}^{(q)} = \mathbf{t}^{(q)}, q = 1, 2, \ldots, Q$, and therefore $M = N$. Thus the weight matrix is an N by N dimensional matrix. We will use bipolar neurons and the Hebb's rule to train the NN, assuming zero initial weights. We will also assume that the NN has zero biases.

Sometimes one also assumes that the autoassociative net has no self-connections. This means that for $i = 1, 2, \ldots, N$, input neuron X_i has no connection with output neuron Y_i. In other words, the diagonal elements of the weight matrix are set to zeroes.

Let us take $N = 4$ and consider storing just one vector

$$\mathbf{s} = \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}.$$

Hebb rule gives the following weight matrix

$$\begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

With just one training vector, it is clear that the NN can store that pattern. Putting pattern **s** in the NN produces exactly the same pattern as its output, since there is absolutely no cross talk in this case.

It is more interesting to consider introducing imperfections in the input data. We can introduce a single mistake in the input vector by flipping one of the 4 bits. One finds that the original pattern can be recalled perfectly (total recall). We can also introduce 2 missing entries in the pattern by replacing any two of the bits by 0. One finds perfect recall also. However if 2 or more mistakes are introduced any where in the input pattern, then their output patterns (all given by [0 0 0 0]) are no longer the same as the stored pattern.

Next, we consider storing 2 vectors

$$\mathbf{s}^{(1)} = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix} \qquad \mathbf{s}^{(2)} = \begin{bmatrix} -1 & 1 & 1 & -1 \end{bmatrix}.$$

Notice that these vectors are orthogonal to each other. One obtains from Hebb rule the following weight matrix

$$\begin{bmatrix} 2 & 0 & -2 & 0 \\ 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 \\ 0 & -2 & 0 & 2 \end{bmatrix},$$

or

$$\begin{bmatrix} 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \\ -2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \end{bmatrix},$$

if the diagonal elements of the weight matrix are set to zero. One can verify that both of the original patterns can be recalled in either cases.

Next, we consider storing 2 non-orthogonal vectors

$$\mathbf{s}^{(1)} = \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}, \qquad \mathbf{s}^{(2)} = \begin{bmatrix} 1 & 1 & -1 & 1 \end{bmatrix}.$$

One finds that neither of these 2 vectors can be recalled perfectly if we zero out the diagonal elements of the weight matrix. (If the diagonal elements are kept, then both patterns can be recalled perfectly.) The output vector in either case is $\begin{bmatrix} 1 & 0 & -1 & 1 \end{bmatrix}$, with an uncertain second element. This is precisely the element that distinguishes between the 2 original vectors. Thus the NN has done as much as it can, given the similarities of the 2 stored vectors.

One finds that the NN can store the following 3 mutually orthogonal vectors

$$\mathbf{s}^{(1)} = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}, \quad \mathbf{s}^{(2)} = \begin{bmatrix} -1 & 1 & 1 & -1 \end{bmatrix}, \quad \mathbf{s}^{(3)} = \begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix},$$

as expected. This is true whether or not the diagonal elements of the weight matrix are set to zero.

If we try to store one more vector by adding

$$\mathbf{s}^{(4)} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

to the training set, we find that the weight matrix becomes a zero matrix. Thus the NN cannot recall any of the training vectors. Notice that all 4 training vectors are mutually orthogonal (that is they are as distinct as can be).

5. Capacity of an Autoassociative NN

The capacity of an autoassociative NN is defined to be the maximum number of patterns that can be stored.

The capacity of an autoassociative NN depends on

1. the number of components, N, of each of the stored vectors. The larger N is, the higher is the capacity.
2. the relationship among the stored vectors. The more uncorrelated they are, the higher is the capacity.

We have the following theorem proved by Szu concerning the capacity of an autoassociative NN with bipolar neurons.

Szu proved that no more than $N-1$ mutually orthogonal bipolar vectors, each with N (even) components, can be stored using the Hebb rule for the weights if the diagonal terms are set to zero.

If the patterns are $\mathbf{s}^{(q)}, q = 1, 2, \ldots, Q$, then Hebb rules gives the weight matrix

$$w_{ij} = \begin{cases} 0, & \text{if } i = j \\ \sum_{q=1}^{Q} s_i^{(q)} s_j^{(q)}, & \text{if } i \neq j. \end{cases}$$

We want to know if the NN can recall $\mathbf{s}^{(p)}$ if p is one of the stored vectors. With $\mathbf{s}^{(p)}$ as the input, we have

$$y_{\text{in,j}} = \sum_{i=1}^{N} s_i^{(p)} w_{ij} = \sum_{i \neq j}^{N} s_i^{(p)} \sum_{q=1}^{Q} s_i^{(q)} s_j^{(q)} = \sum_{q=1}^{Q} s_j^{(q)} \sum_{i \neq j}^{N} s_i^{(p)} s_i^{(q)}.$$

By assumption, the stored vectors are mutually orthogonal and there-

fore

$$\sum_{i=1}^{N} s_i^{(p)} s_i^{(q)} = 0, \qquad \text{if } p \neq q,$$

and

$$\sum_{i=1}^{N} s_i^{(p)} s_i^{(q)} = N, \qquad \text{if } p = q.$$

Therefore

$$\sum_{i \neq j}^{N} s_i^{(p)} s_i^{(q)} = -s_j^{(p)} s_j^{(q)}, \qquad \text{if } p \neq q,$$

and

$$\sum_{i \neq j}^{N} s_i^{(p)} s_i^{(q)} = N - 1, \qquad \text{if } p = q.$$

Consequently

$$y_{\text{in},j} = (N-1)s_j^{(p)} - \sum_{q \neq p}^{Q} s_j^{(p)}[s_j^{(q)}]^2 = (N-Q)s_j^{(p)}.$$

If the number of stored vectors Q is less than N, then $(N-Q) > 0$, and so

$$y_j = s_j^{(p)}.$$

This means that we can store no more than N vectors. If $Q = N$, then $y_{\text{in},j} = 0$, and so $y_j = 0$ (output components are all uncertain). Of course we cannot have $Q > N$, since in N dimensions, there cannot be more than N mutually orthogonal vectors.

Notice that the dot-product of any 2 bipolar vectors cannot be zero unless N is even. Therefore the above theorem applies only if N is even, otherwise we do not have any orthogonal vectors at all.

We also have the following theorem:

A set of 2^k mutually orthogonal bipolar vectors can be constructed for $N = 2^k m$ with m odd, and no larger set can be formed.

First some observations:

1. By the fundamental theorem of arithmetic, every positive integer such as N has a unique prime factorization:

   $$N = p_1 p_2 \dots p_\ell.$$

   Of all the prime numbers, 2 is the only one that is even. Suppose out of the ℓ prime factors, the number 2 appears k times, and so we know that N has a factor 2^k. The remaining prime factors are all odd and therefore their product m is also odd. Thus any positive integer can be expressed uniquely as $2^k m$, for a non-negative integer k, and an odd positive integer m.

2. If $\mathbf{v}$ is a vector of a certain length, then $\begin{bmatrix} \mathbf{v} & \mathbf{v} \end{bmatrix}$, the concatenation of $\mathbf{v}$ with itself is a vector having twice the original length.

3. If **a** and **b** are 2 mutually orthogonal bipolar vectors of the same length, then $\left[\begin{array}{cc}\mathbf{a} & \mathbf{a}\end{array}\right]$, $\left[\begin{array}{cc}\mathbf{a} & -\mathbf{a}\end{array}\right]$, $\left[\begin{array}{cc}\mathbf{b} & \mathbf{b}\end{array}\right]$, $\left[\begin{array}{cc}\mathbf{b} & -\mathbf{b}\end{array}\right]$, are 4 mutually orthogonal vectors, each having twice the original length.

4. It is not possible to construct a pair of orthogonal bipolar vectors if N is odd (*i.e.* $k = 0$), since the dot-product of any two bipolar N-dimensional vectors must be an odd integer and therefore cannot be 0.

This theorem is proved by explicitly constructing these vectors for a given N. N can be written uniquely as $2^k m$, for a non-negative integer k, and an odd positive integer m. We first start with an m dimensional bipolar vector $\mathbf{v}_m(1) = \left[\begin{array}{cccc}1 & 1 & \ldots & 1\end{array}\right]$. In step $i = 1$, we can construct from $\mathbf{v}_m(1)$ 2^i $(= 2)$ orthogonal $2m$ dimensional bipolar vectors

$$\mathbf{v}_{2m}(1) = \left[\begin{array}{cc}\mathbf{v}_m(1) & \mathbf{v}_m(1)\end{array}\right]$$

and

$$\mathbf{v}_{2m}(2) = \left[\begin{array}{cc}\mathbf{v}_m(1) & -\mathbf{v}_m(1)\end{array}\right].$$

The subscript is used to denote the length of the vectors. The number inside the parentheses labels the vector constructed at a given step.

One can show that there is no other $2m$ dimensional bipolar vector that is orthogonal to both $\mathbf{v}_{2m}(1)$ and $\mathbf{v}_{2m}(2)$. To see that, let us assume that such a vector exists and write it in the form $\begin{bmatrix} \mathbf{a}_m & \mathbf{b}_m \end{bmatrix}$, where $\mathbf{a}_m$ and $\mathbf{b}_m$ are each m dimensional bipolar vectors. The orthogonality with both $\mathbf{v}_{2m}(1)$ and $\mathbf{v}_{2m}(2)$ means that

$$\mathbf{a}_m \cdot \mathbf{v}_m(1) + \mathbf{b}_m \cdot \mathbf{v}_m(1) = 0,$$

and

$$\mathbf{a}_m \cdot \mathbf{v}_m(1) - \mathbf{b}_m \cdot \mathbf{v}_m(1) = 0,$$

These equations imply that vectors $\mathbf{a}_m$ and $\mathbf{b}_m$ obey

$$\mathbf{a}_m \cdot \mathbf{v}_m(1) = 0 = \mathbf{b}_m \cdot \mathbf{v}_m(1).$$

However since m is odd, it is clear that it impossible to find such vectors.

In step $i = 2$, we construct the following 4 $(= 2^i = 2^2)$ mutually

orthogonal $4m$ dimensional vectors:

$$\mathbf{v}_{4m}(1) = \left[\begin{array}{cc} \mathbf{v}_{2m}(1) & \mathbf{v}_{2m}(1) \end{array} \right],$$

$$\mathbf{v}_{4m}(2) = \left[\begin{array}{cc} \mathbf{v}_{2m}(1) & -\mathbf{v}_{2m}(1) \end{array} \right],$$

$$\mathbf{v}_{4m}(3) = \left[\begin{array}{cc} \mathbf{v}_{2m}(2) & \mathbf{v}_{2m}(2) \end{array} \right],$$

and

$$\mathbf{v}_{4m}(4) = \left[\begin{array}{cc} \mathbf{v}_{2m}(2) & -\mathbf{v}_{2m}(2) \end{array} \right].$$

At each step, the number of mutually orthogonal vectors doubles, and the length of each of these vectors also doubles. By using basically the same argument that we used in step 1, it is easy to see that it is impossible to have in the i-th step a bipolar vector of length $2^i m$ that is orthogonal to all the 2^i mutually orthogonal vectors, $\mathbf{v}_{2^i m}(1)$, $\mathbf{v}_{2^i m}(2)$, ..., $\mathbf{v}_{2^i m}(2^i)$.

We can continue this way until at the k-th step we have exactly 2^k mutually orthogonal bipolar vectors, $\mathbf{v}_N(1)$, $\mathbf{v}_N(2)$, ..., $\mathbf{v}_N(2^k)$.

References

[1] See Chapter 3 in Laurene Fausett, "Fundamentals of Neural Networks - Architectures, Algorithms, and Applications", Prentice Hall, 1994.