POLYTECHNIC UNIVERSITY
Department of Computer and Information Science

Fixed Weight Competitive Nets: Hamming Net

K. Ming Leung

Abstract: A fixed weight competitive net known as the Hamming net is discussed.

Directory

Table of Contents

1. Introduction

When a net is trained to classify the input signal into one of the output categories, A, B, C, D, E, J, or K, the net sometimes responded that the signal was both a C and a K, or both an E and a K, or both a J and a K, due to similarities in these character pairs. In this case it will be better to include additional structure in the net to force it to make a definitive decision. The mechanism by which this can be accomplished is called competition.

The most extreme form of competition among a group of neurons is called Winner-Take-All, where only one neuron (the winner) in the group will have a nonzero output signal when the competition is completed. An example of that is the MAXNET. A more general form of competition useful for contrast enhancement is known as the Mexican Hat. A simple clustering net known as the Hamming net is discussed here. It is based on the use of fixed exemplar vectors and the MAXNET subnet.

2. MAXNET

MAXNET is a neural net based on competition that can be used as a subnet to choose the neuron whose activation is the largest. If a fully neural implementation is not required, one can certainly use any of a multitude of algorithms that find the maximum value of a set of numbers.

The MAXNET has M neurons that are fully interconnected, each neuron is connected to every other neuron in the net, including itself. The transfer function used by the neurons is the positive-linear function

$$f_{\text{poslin}}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0. \end{cases}$$

There is no training for the MAXNET. The weights are symmetrical, fixed and are given by

$$w_{ij} = \begin{cases} 1, & \text{if } i = j \\ -\epsilon, & \text{if } i \neq j, \end{cases}$$

where, according to our textbook,

$$0 < \epsilon < \frac{1}{M}$$

is a predetermined positive constant. It has to be positive and smaller than 1. The value of its upper bound will be determined later. From the diagonal terms in the weight matrix, we see that each neuron is connected to itself with a positive weight of 1. This represents self-promotion in a competitive environment. The off-diagonal terms, $-\epsilon$, is negative and thus represent inhibition. These terms represent the suppression of the signals of all the other neurons. Self-promoting ones own achievements while suppressing those of ones peers are commonly found in highly competitive sociological environments since every one wants to be a winner. Notice that since $\epsilon < 1$, self-promotion is more important than suppressing the signals of other neurons.

The activation of the neurons are specified by $a_j, j = 1, \ldots, M$, and they must all be non-negative for the MAXNET to work. The MAXNET algorithm is:

1 Choose an ϵ for the weight matrix (but no need to set it up), and initialize the activations $a_j(0), j = 1, \ldots, M$ (to non-negative values).

2 For $k = 1, 2, \ldots$ repeat steps a - c while stopping condition is not met.

  a For each neuron $i = 1, \ldots, M$, compute the net signal it receives for the next step

$$a_{\text{in},i}(k) = a_i(k-1) - \epsilon \sum_{j \neq i} a_j(k-1).$$

  b Update the activations for $i = 1, \ldots, M$

$$a_i(k) = f_{\text{poslin}}(a_{\text{in},i}(k)).$$

  c Test stopping condition: Stop only when only one activation is nonzero.

We assume here that in the real world, only one neuron, not two or more neurons, can have the same maximal activation, otherwise special precautions must be exercised to handle the situation.

Since the activations are all non-negative, it is clear that for all i $a_{\text{in},i}(k) \leq a_i(k-1)$, and so as MAXNET iterates, activations of all the neurons decrease. However, the smaller their activations are, the more they decrease fractionally. (The scheme hurts the poor more than the rich.) As the net iterates, neurons with the smallest values of $a_{\text{in},i}(k)$ are driven to negative first. Their transfer functions then yield zero values for their activations. Once the activation is driven to zero, it is clear that it will remain at zero with subsequent iterations. Until eventually the activities of all the neurons except one, the winner, are driven to zero. The activation for the winner then ceases to decrease any further.

Given a group of neurons, the MAXNET is often used in competitive nets as a subnet to find the one that has the largest activation. With suitable modifications, it can also be used to determine the minimum of a set of given quantities. The operation of such a subnet is often encapsulated as a vector-valued function

$$\mathbf{y} = \text{compet}(\mathbf{x}),$$

where **x** is the input and **y** is the corresponding output of the subnet. There should only be one positive element in **y**, all other element must be identically zero.

2.1. Example: MAXNET

The following example shows how the action of the MAXNET for the case of $M = 4$ neurons with initial activations $a_1(0) = 0.6$, $a_2(0) = 0.2$, $a_3(0) = 0.8$, $a_4(0) = 0.4$. We choose $\epsilon = 0.2$. Since $1/M = 0.25$, the chosen value for ϵ is within the required range.

step k	$a_1(k)$	$a_2(k)$	$a_3(k)$	$a_4(k)$
0	0.6	0.2	0.8	0.4
1	0.32	0	0.56	0.08
2	0.192	0	0.48	0
3	0.096	0	0.442	0
4	0.008	0	0.422	0
5	0	0	0.421	0

In increasing order of their activations, the neurons are a_2, a_4, a_1, and a_3. In the first step, their activation decrease fractionally by 1, 0.8, 0.47, and 0.3, respectively. The scheme hurts the poor much more so than for the rich. They are driven to zero also in this order until only one winner survives.

The larger the value of ϵ is, the faster the activations of the losing neurons are driven to zero. For example, if we choose $\epsilon = 0.3$, then we obtain the following results.

step k	$a_1(k)$	$a_2(k)$	$a_3(k)$	$a_4(k)$
0	0.6	0.2	0.8	0.4
1	0.18	0	0.44	0
2	0.048	0	0.386	0
3	0	0	0.372	0

The winner emerges only after three iterations. If ϵ is in $[3/7\ 2/3)$, then only one iteration is needed to obtain a winner. For $\epsilon \geq 2/3$, no winner can be found since all the activations are driven to zero in one single step. Clearly the question is how large a value can one use for ϵ for fast convergence?

2.2. Best Choice of ϵ

We want to determine a good choice of ϵ for fast convergence.[2] According to our textbook

$$0 < \epsilon < \frac{1}{M}.$$

We want to see if this is true, and how the upper bound for the value of ϵ comes about.

The fastest convergence can be achieved if one can choose an ϵ such that the activations of all neurons except the winning one are driven to zero in one iteration. Since the signal receives by a given neuron, say i, in step k is

$$a_{\text{in},i}(k) = a_i(k-1) - \epsilon \sum_{j \neq i} a_j(k-1),$$

and *if* we *knew* that neuron ℓ has the largest activation, then by choosing ϵ to be slightly less than

$$\epsilon_{\max} = \frac{a_\ell}{\sum_{j \neq \ell}^{M} a_j} = \frac{1}{\sum_{j \neq \ell}^{M} \frac{a_j}{a_\ell}}$$

then in a single iteration, $a_{\text{in},\ell}(k)$ becomes only slightly larger than zero and therefore so is a_ℓ. This means that all the other $a_{\text{in},i}(k)$ becomes negative, and so a_i becomes zero because of the transfer function. Of course we do not know which of the neurons has the largest activation, finding that is in fact the problem we are confronted with. So we cannot possibly know how large $\epsilon_{\max}$ is. We will replace that by a smaller number. As a result we will need to iterate MAXNET a few times before convergence. This smaller number is obtained from the above expression by replacing the denominator with a larger number. Notice that $\frac{a_j}{a_\ell} \leq 1$, and if we replace that by 1, the denominator become $M - 1$. Thus we want to choose

$$\epsilon = \frac{1}{M-1}.$$

This value is larger than the one suggested in our textbook, and leads to a slightly faster convergence especially when M is not too large[2].

3. Mexican Hat Network

Mexican Hat Network is good for the purpose of contrast enhancement. Some common topological arrangements of the neurons include linear and two-dimensional geometries. Parameter R_1 specifies the radius of the region with positive reinforcement. Parameter R_2 ($> R_1$) specifies the radius of the region of interconnections. The weights are determined by

$$w_k = \begin{cases} c_1 > 0, & \text{for } |k| \le R_1 \\ c_2 < 0, & \text{for } R_1 < |k| \le R_2. \end{cases}$$

The transfer function used here is defined by

$$f_{\text{satlin}}(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } 0 \le x \le x_{\max} \\ x_{\max}, & \text{if } x \ge x_{\max} \end{cases}$$

which can also be expressed as

$$f_{\text{satlin}}(x) = \min(x_{\max}, \max(0, x)) = \max(0, \min(x_{\max}, x)).$$

where $x_{\max}$ is fixed positive quantity.

The Mexican Hat net has parameters, R_1, R_2, c_1, c_2, $x_{\max}$ and the maximum number of iteration, $t_{\max}$. These are all model specific and must be set at the outset.

The algorithm is:

1 Set parameters, R_1, R_2, c_1, c_2, $x_{\max}$ and $t_{\max}$. Set the weights

$$w_k = \begin{cases} c_1 > 0, & \text{for } |k| \leq R_1 \\ c_2 < 0, & \text{for } R_1 < |k| \leq R_2. \end{cases}$$

2 Let $\mathbf{x}(0) = \mathbf{s}$.

3 For $t = 1, \ldots, t_{\max}$ repeat steps a - b.

a For each neuron $i = 1, \ldots, N$, compute the net input

$$\begin{aligned} x_{\text{in},i}(t) &= c_1 \sum_{k=-R_1}^{R_1} x_{i+k}(t-1) + c_2 \sum_{k=-R_2}^{-R_1-1} x_{i+k}(t-1) \\ &\quad + c_2 \sum_{k=R_1+1}^{R_2} x_{i+k}(t-1) \end{aligned}$$

b Apply the transfer function for $i = 1, \ldots, N$

$$x_i(t) = f_{\text{satlin}}(x_{\text{in},i}(t)).$$

4. Hamming Net

For the Hamming net, M exemplar bipolar vectors $\mathbf{e}^{(1)}$, $\mathbf{e}^{(2)}$, . . ., $\mathbf{e}^{(M)}$ are given. They are used to form the weight matrix of the network. The Hamming net is therefore a fixed-weight net. For any given input vector $\mathbf{x}$, the Hamming net finds the exemplar vector that is closest to $\mathbf{x}$. Therefore if a collection of input vectors are given, the Hamming net can be used to cluster these vectors into M different groups.

We need to define what we mean by saying that two vectors $\mathbf{a}$ and $\mathbf{b}$ are close to each other. Again we will be working with bipolar vectors and denote the Hamming distance between the vectors (defined to be the number of corresponding bits that are different between the vectors) by $H(\mathbf{a}, \mathbf{b})$, and the number of corresponding bits that agrees with each other by $A(\mathbf{a}, \mathbf{b})$.

It is clear that for bipolar vectors

$$\mathbf{a} \cdot \mathbf{b} = A(\mathbf{a}, \mathbf{b}) - H(\mathbf{a}, \mathbf{b}),$$

because the corresponding bits are either the same, and so they contribute 1 to the dot-product, or they are different and so contribute -1 to the dot-product. A corresponding pair of bits must either agree or disagree, and so

$$N = A(\mathbf{a}, \mathbf{b}) + H(\mathbf{a}, \mathbf{b}).$$

We eliminate the Hamming between these two equations and solve for $A(\mathbf{a}, \mathbf{b})$ to get

$$A(\mathbf{a}, \mathbf{b}) = \mathbf{a} \cdot \frac{\mathbf{b}}{2} + \frac{N}{2}.$$

The two vectors **a** and **b** are close to each other if $A(\mathbf{a}, \mathbf{b})$ is large.

We will use $A(\mathbf{a}, \mathbf{b})$ to measure the closeness of two vectors in the Hamming net. The closest two vectors are to each other the larger is $A(\mathbf{a}, \mathbf{b})$. We need to work with positive quantities that are large when two vectors are close to each other because MAXNET will be used as a subnet to find the one exemplar vector that is closest to the given input vector. Therefore we can only work with $A(\mathbf{a}, \mathbf{b})$ rather than $\mathbf{a} \cdot \mathbf{b}$ or $H(\mathbf{a}, \mathbf{b})$.

We want to identify **a** with an input vector **x** (assumed to be a row vector of length N), and **b** with one of the exemplar vectors. With M exemplar vectors, they can be scaled down by a half and stored column-wise to form the weight matrix for the Hamming net. Thus the weight matrix is $N \times M$, and its elements are

$$w_{ij} = \frac{1}{2} e_i^{(j)}, \qquad i = 1, \dots, N, \quad \text{and} \quad j = 1, \dots, M.$$

The bias vector is a row vector of length M and has elements $N/2$:

$$b_j = \frac{1}{2} N, \quad j = 1, \dots, M.$$

For a given input vector $\mathbf{x}$, the number of bits of agreement that it has with each of the exemplar vector form a vector

$$\mathbf{y}_{\text{in}} = \mathbf{x}\mathbf{W} + \mathbf{b},$$

of inputs to the neurons at the Y-layer. These neurons have identity transfer functions and send the signals that they received to a MAXNET to find the one with the largest agreement.

The algorithm of the Hamming net is:

1 Set the weights and biases

$$w_{ij} = \frac{1}{2} e_i^{(j)}, \qquad i = 1, \ldots, N, \quad \text{and} \quad j = 1, \ldots, M,$$

$$b_j = \frac{1}{2} N, \quad j = 1, \ldots, M.$$

2 For a given input vector $\mathbf{x}$, repeat steps a - c.

a Compute the input to each unit $j = 1, \ldots, M$

$$y_{\text{in},j} = \sum_{i=1}^{N} x_i w_{ij} + b_j.$$

b Initialize the activations for MAXNET

$$y_j(0) = y_{\text{in},j}, \quad \text{for} \quad j = 1, \ldots, M.$$

c MAXNET finds the best match exemplar vector for the given input vector.

4.1. Example: Hamming Net

We consider here an example of a Hamming net that has the following two exemplar vectors:

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

Notice that these two exemplar vector are orthogonal to each other. We are given the following four vectors

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

$$\mathbf{x}^{(3)} = \begin{bmatrix} -1 & -1 & -1 & 1 \end{bmatrix}, \qquad \mathbf{x}^{(4)} = \begin{bmatrix} -1 & -1 & 1 & 1 \end{bmatrix}.$$

We will use the Hamming net to cluster them into two groups, one group for each of the exemplar vectors. Thus for each of the input vectors we need to find the exemplar vector that is closest to it.

First we set up the weights and biases:

$$\mathbf{W} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 2 & 2 \end{bmatrix}.$$

For $\mathbf{x}^{(1)}$, we find that

$$\mathbf{y}_{\text{in}} = \mathbf{x}^{(1)}\mathbf{W} + \mathbf{b} = \begin{bmatrix} 3 & 1 \end{bmatrix}.$$

With this as input to MAXNET, one finds that the exemplar vector that is the closest to input $\mathbf{x}^{(1)}$ is $\mathbf{e}^{(1)}$. This makes sense since $A(\mathbf{x}, \mathbf{e}^{(1)}) = 3$ and $A(\mathbf{x}, \mathbf{e}^{(2)}) = 1$. We repeat the procedure for $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$ and $\mathbf{x}^{(4)}$, and find that $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are closest to $\mathbf{e}^{(1)}$, and $\mathbf{x}^{(3)}$ and $\mathbf{x}^{(4)}$ are closest to $\mathbf{e}^{(2)}$. Thus these four input vectors are separated into two separate clusters.

References

[1] See Chapter 4 in Laurene Fausett, "Fundamentals of Neural Networks - Architectures, Algorithms, and Applications", Prentice Hall, 1994.

[2] For a faster converging MAXNET, see for example, J.-C. Yen and S. Chang, "Improved Winner-Take-All Neural Network", Electronics Letters, Vol.28, No.7 662-664, 1992..

10, 11