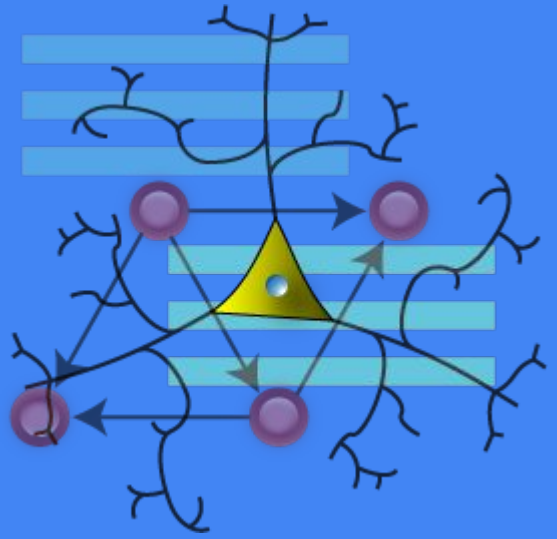
HAMMING NET



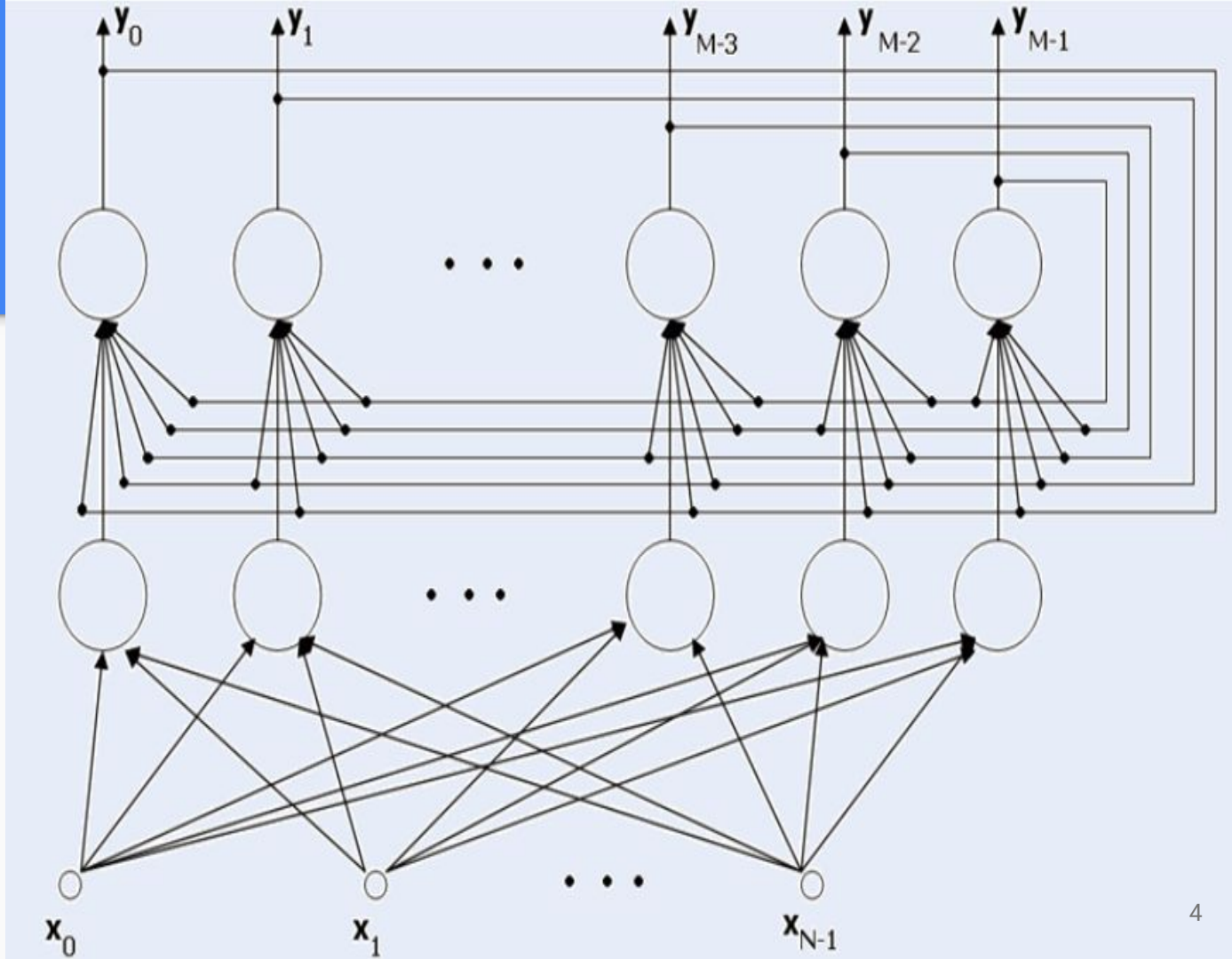
HAMMING NET

Kundan Shumsher Rana (Introduction)
Anju Shahi (Structure)
Surya Raj Timsina (How it works?)
Sagar Giri (Example)

Introduction

- **Hamming net**: a Hamming distance based neural network.
- Hamming net is a double layer neural network.
- It is the neural network implementation of the Hamming distance based NNC (Nearest Neighbour Classifier).
- The inputs are binary numbers {0,1} or {-1,1}.
- We consider only bi-polar case here.
- The outputs are the similarities between the input pattern and the weight vectors of the neurons.

Structure



Structure (Explanation)

- In the picture presented above we can see the Hamming Network. It can be divided into two basic sections:
- **input layer** – a layer built with neurons, all of those neurons are connected to all of the network inputs;
- **output layer** – which is called MaxNet layer; the output of each neuron of this layer is connected to input of
- each neuron of this layer, besides every neuron of this layer is connected to exactly one neuron of the input layer (as in the picture above).
- It’s easy to see, that both layers have the same number of neurons.

How does the Hamming Network work?

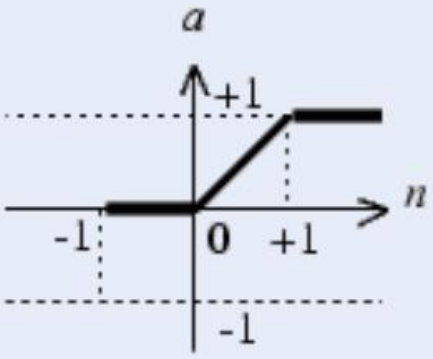
- Input layer neurons are programmed to identify a fixed number of patterns; the number of neurons in this layer matches the number of those patterns (M neurons – M patterns).
- Outputs of these neurons realise the function, which “measures” the similarity of an input signal to a given pattern.

How does the Hamming Network work?

- The output layer is responsible for choosing the pattern, which is the most similar to testing signal.
- In this layer, the neuron with the strongest response stops the other neurons responses (it usually happens after few calculating cycles, and there must be a “0” on x(i) inputs during these cycles).

How does the Hamming Network work? (contd..)

There should be the “1 of M” code used on the output, which points at the network answer (1 of M patterns is recognized). To achieve this, proper transfer function has to be used – in our case the best function was...



$a = satlin(n)$

Example - (application in character recognition)

- The task is to recognize 1 of 8 signs recorded in the network “memory”.
- The picture below shows 8 used signs:

- The matrices that represent these signs are two- dimensional (3x5 pixels).
- Black squares are ones, white squares are zeros.
- To make proper calculations, we normalised all of the row vectors.

Example - (application in character recognition)

- We want to classify 8 signs, so our network will contain 16 neurons, divided into 2 groups, 8 neurons in each layer.
- Weights of the input layer are predefined as transposed matrix of signs – previously normalised.
- Weights of the output layer are defined according to the previously stated assumptions and the ‘–1/M’ parameter in our case was set to ‘–1/8’.
- We cannot forget that it is necessary to normalise rows of that matrix.

Example - (application in character recognition)

- We assumed that 15 calculating cycles in the output layer will be sufficient to stabilize the network response.
- But we noticed that it can be not enough, when we are dealing with strong noise.
- It would be the best solution to check if there is a “1” on exactly one of the outputs each cycle, but it would be difficult to predict the amount of time needed to perform the classification -- and that is very important when working with DSP.

Example - (application in character recognition)

- The network created works correctly – it recognises given sign, even with noise, if only it is possible.
- It does not recognise properly when it is totally impossible to do it or when 2 neurons have strong responses, similar to each other – it would take more calculating cycles to recognise it, then.

Sources

- http://web-ext.u-aizu.ac.jp/~qf-zhao/TEACHING/NN-I/Lec05-1.pdf
- http://cis.poly.edu/~mleung/CS6673/s09/HammingNet.pdf
- http://home.agh.edu.pl/~vlsi/AI/hamming_en/