Distance Based Neural Networks - 1

Contents of this lecture

- Nearest neighbor classifier (NNC).
- Hamming net: a Hamming distance based neural network.
- Max-Net: a neural network for finding the maximum value.
- Kohonen net: a similarity based neural network.
- Winner-take-all learning.

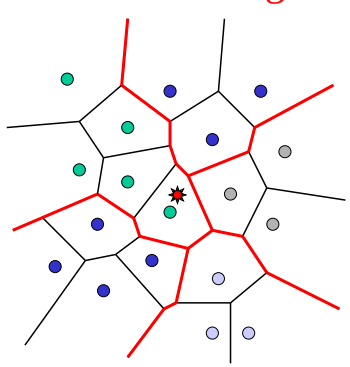
NNC: Nearest Neighbor Classifier

- NNC is a simple method for pattern recognition.
- Suppose that we have *p* prototypes with known class labels.
- For any given pattern *x*, it is assigned to the class label of the *i-th* prototype if

$$i = \arg\min_{k} \text{distance}(x, y_k)$$

- Examples of distance measures include the Hamming distance, Euclidean distance, and Mahalanobis distance.

Nearest neighbor classifier（NNC)



- An NNC contains a set of prototypes.
- For any unknown data point, it is classified to the same class as the nearest neighbor.

Important features of the NNC

- If the number of prototypes is large enough, the classification error will be less than $2E$, where E is the theoretically minimum classification error.
- The system can be built-up simply by inserting (memorizing) new patterns into the system.
- A large number of patterns must be used

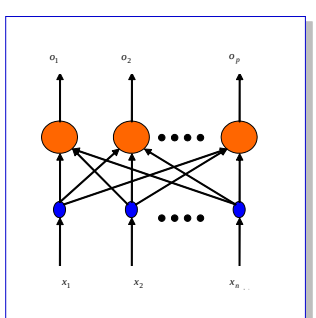
Quick for learning, but slow for recalling

Hamming net

- Hamming net is a single layer neural network.
- **It is the neural network implementation of the Hamming distance based NNC.**
- 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 of Hamming net

- The number of inputs is n, which is the dimensionality of the pattern space.
- The number of outputs is p, which is the number of patterns to store.



Relationship between Hamming distance and similarity

The Hamming distance between two bipolar binary vectors is

$HD(x,y)$ = number of different bits

On the other hand, the inner product or similarity between x and y is

$$< x,y >= x^t y = (n - HD(x,y)) - HD(x,y)$$

or equivalent ly,

$$\frac{1}{2} < x, y >= \frac{n}{2} - HD(x,y)$$

To find the weights of Hamming net

Suppose that the patterns to be stored are given by $s^{(1)}, s^{(2)},..., s^{(p)}$, these patterns can be stored in the Hamming - net by choosing the weights as follows :

$$W_H = \frac{1}{2}\begin{bmatrix} s_1^{(1)} & s_2^{(1)} & \cdots & s_n^{(1)} \\ s_1^{(2)} & s_2^{(2)} & \cdots & s_n^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ s_1^{(p)} & s_2^{(p)} & \cdots & s_n^{(p)} \end{bmatrix}$$

To find the output

The bias for each neuron is fixed to $n/2$, and the effective input of the ***m-th*** neuron is given by

$$net_m = \sum_{i=1}^{n} w_{mi} x_i + \frac{n}{2} = \frac{1}{2} x^t s^{(m)} + \frac{n}{2} = n - HD(x, s^{(m)})$$

The final output is calculated by

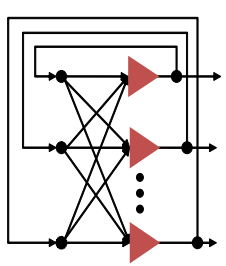
$$f(net_m) = \frac{net_m}{n} = 1 - \frac{HD(x, s^{(m)})}{n}$$

This is actually the normalized similarity between the weight vector of the ***m*** - th neuron and the input ***x***.

The similarity is one when the distance is zero.

MAXNET: A Neural Network for Finding the Maximum Value

- MAXNET is similar to HNN.
- It is a single layer feed back neural network with *p* neurons.
- The diagonal elements of the weight matrix is always 1, and all other elements are –ε.
- The parameter ε is a constant in [0,1/p).
- The inputs (initial states) are real numbers in [0,1].



To find the output of MAXNET

- All neurons are updated in synchronous mode.
- The effective input of the *i*-th neuron is calculated by

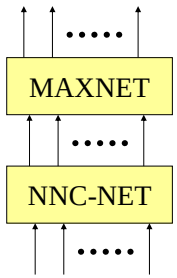
$$net_i = x_i^{old} - \sum_{\substack{j=1 \\ j \neq i}}^{p} \varepsilon \cdot x_j^{old}$$

- and the final output is given by

$$o_i = f(net_i) = \begin{cases} 0, & net_i < 0 \\ net_i, & net_i \geq 0 \end{cases}$$

MAXNET can be used together with the distance based neural networks

- For any input vector, the MAXNET gradually suppresses all but the neuron with the largest initial input.
- Thus, for example, if used with the Hamming net, it can select the prototype that is most similar to the input vector.
- Hamming net finds the similarities between the input pattern and the weight vectors of all neurons.
- And the most active neuron is selected by MAXNET, and is used as the final output.



Example 1

- There are 3 patterns to be stored in the network
  - S(1)=[1 1 1 1 -1 -1 1 1 1]^t
  - S(2)=[-1 1 -1 -1 1 -1 -1 1 -1]^t
  - S(3)=[1 1 1 -1 1 -1 -1 1 -1]^t
- The weight matrix for the Hamming net is given on the right.
- The parameter ε in the weight matrix for the MAXNET is given by ε=0.2<1/3.

$$W_H = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \end{bmatrix}$$

Test the first pattern

Output of the Hamming net is: 1.000 0.333 0.556

State transition of MAXNET:

0.822 0.022 0.289
0.760 0.000 0.120
0.736 0.000 0.000

The current input is the 1st pattern

Test the second and the third patterns

- Output of the Hamming net is:
- 0.333 1.000 0.778
- State transition of MAXNET:
- 0.000 0.778 0.511
- 0.000 0.676 0.356
- 0.000 0.604 0.220
- 0.000 0.560 0.100
- 0.000 0.540 0.000
- The current input is the 2nd pattern.

- Output of the Hamming net is: 0.556 0.778 1.000
- State transition of MAXNET:
- 0.200 0.467 0.733
- 0.000 0.280 0.600
- 0.000 0.160 0.544
- 0.000 0.051 0.512
- 0.000 0.000 0.502
- The current input is the 3rd pattern.

Neural networks based on the Euclidean distance

- If the patterns to be stored are *n*-dimensional real vectors, Hamming distance cannot be used.
- The Euclidean distance between two vectors are defined by

$$\|\boldsymbol{x}-\boldsymbol{y}\| = \sqrt{(\boldsymbol{x}-\boldsymbol{y})^{\mathrm{t}}(\boldsymbol{x}-\boldsymbol{y})} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

A special case

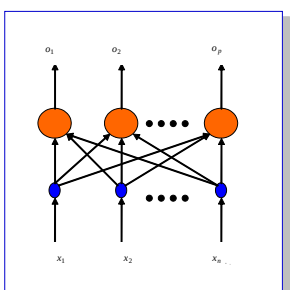
If the norm of the vectors are normalized, such that $\|\boldsymbol{x}\| = (\boldsymbol{x}^{\mathrm{t}}\boldsymbol{x})^{-2} = 1$, then, the similarity between two vectors can be defined by

$$< x,y >= \boldsymbol{x}^{\mathrm{t}}\boldsymbol{y} = \sum_{i=1}^{n} x_i y_i$$

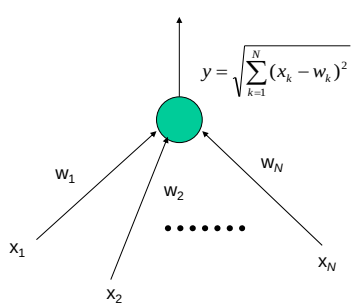
We say two vectors are close to each other if the distance between them is small, or for the normalized vectors, if their similarity is high.

The Kohonen neural network

- The right figure shows a neural network based on the Euclidean distance.
- This network is also called the Kohonen self-organizing neural network.
- There are *p* prototype neurons, and *p* can be smaller than the total number of patterns to be stored.



Structure of a distance neuron



This neuron is not a switch. It simply provides information for making a decision for the next stage

Clustering of patterns

- Suppose that we have P patterns.
- Instead of remembering all these patterns, we can group them into p ($p << P$) clusters, and find one representative or prototype for each cluster.
- Each neuron in the Kohonen neural network is a prototype.
- It is an abstracted pattern representing many realistic patterns (e.g. face, chair).
- The weights of the neurons are found through learning.

Winner-take-all learning

Step 1: Initialize all weights with random numbers.

Step 2: Identify the winner

$$\forall x \in \text{Training set}: \; m = \arg\min_{i=1,2,\ldots,p} \|x - w_i\|$$

the mth neuron is called the winner. Here, w_i is the weight vector of the ith neuron.

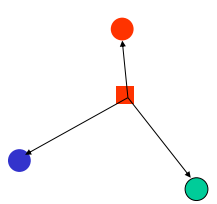
Step 3: Update the weights of the winner

$$w_m^{k+1} = w_m^k + \alpha(x - w_m^k)$$

where α is a small positive constant, and is called the learning rate. Usually, $\alpha \in [0.1, 0.7]$.

Important remarks (1)

- The physical meaning of the winner-take-all learning algorithm is that, when the weight vector of a neuron is close to the input pattern, it is moved towards this pattern, to make them closer.
- The above steps must be performed iteratively for all training data, and the same datum can be re-used.
- Only the weights of the winner is updated for a given input pattern.



Important remarks (2)

- If normalized patterns are considered, the weight vector should be re-normalized after updating.
- The learning rule given here is un-supervised learning.
- Another algorithm called k-means can solve the same problem if all patterns are provided off-line.

Example

- 5 patterns are given as follows:

$$\{x_1, x_2, x_3, x_4, x_5\} = \left\{\begin{bmatrix}0.8\\0.6\end{bmatrix}, \begin{bmatrix}0.1736\\-0.9848\end{bmatrix}, \begin{bmatrix}0.707\\0.707\end{bmatrix}, \begin{bmatrix}0.342\\-0.9397\end{bmatrix}, \begin{bmatrix}0.6\\0.8\end{bmatrix}\right\}$$

- We want to group them into 2 clusters.
- The next slide shows the learning results.
- In this example, all patterns and all weights are normalized.

Results of winner-take-all learning

Results in the 7-th iteration:
0.662825 0.748774
0.286769 -0.958000

Results in the 8-th iteration:
0.662826 0.748774
0.286774 -0.957998

Results in the 9-th iteration:
0.662826 0.748774
0.286775 -0.957998

Results in the 10-th iteration:
0.662826 0.748774
0.286775 -0.957998

Pattern[1] is in 1-th class
Pattern[2] is in 2-th class
Pattern[3] is in 1-th class
Pattern[4] is in 2-th class
Pattern[5] is in 1-th class

$$\begin{bmatrix}0.8\\0.6\end{bmatrix}, \begin{bmatrix}0.1736\\-0.9848\end{bmatrix}, \begin{bmatrix}0.707\\0.707\end{bmatrix}, \begin{bmatrix}0.342\\-0.9397\end{bmatrix}, \begin{bmatrix}0.6\\0.8\end{bmatrix}$$

Centers of the clusters of weights of the neurons

The recalling process

- For any input pattern, it belongs to the *m-th* cluster if the *m-th* neuron is the winner.
- There is no weight updating in the recalling phase.

Team Project IV

- Down load the program for winner-take-all learning from the web page of this course.
- Verify the program using the same data given in example 2.
- Down load the database *Iris* from the UCI Machine Learning Database Repository
  http://www.ics.uci.edu/~mlearn/MLRepository.html
- Modify the program, and test the program using the database Iris. Note that the teacher signals (the last column) of the data are not used in this program. The number of clusters should be determined properly by yourself.

