Artificial Neural Networks

Part 2/3 – Perceptron

Slides modified from Neural Network Design
by Hagan, Demuth and Beale

Berrin Yanikoglu
Perceptron

- A single artificial neuron that computes its weighted input and uses a threshold activation function.

- It effectively separates the input space into two categories by the hyperplane:

  \[ \mathbf{w}^T \mathbf{x} + b = 0 \]
Decision Boundary

The weight vector is orthogonal to the decision boundary.

The weight vector points in the direction of the vector which should produce an output of 1:
- so that the vectors with the positive output are on the right side of the decision boundary
  - if $w$ pointed in the opposite direction, the dot products of all input vectors would have the opposite sign
  - would result in same classification but with opposite labels

The bias determines the position of the boundary:
- solve for $w^T p + b = 0$ using one point on the decision boundary to find $b$. 
Two-Input Case

\[ \mathbf{a} = \text{hardlim}(n) = [1 \ 2] \mathbf{p} + -2 \]

Decision Boundary: all points \( \mathbf{p} \) for which \( \mathbf{w}^T \mathbf{p} + b = 0 \)

If we have the weights and not the bias, we can take a point on the decision boundary, \( \mathbf{p} = [2 \ 0]^T \), and solving for \( [1 \ 2] \mathbf{p} + b = 0 \), we see that \( b = -2 \).
Decision Boundary

\[ w_1^T p + b = 0 \]
\[ w_1^T p = -b \]

- All points on the decision boundary have the same inner product (= -b) with the weight vector
- Therefore they have the same projection onto the weight vector; so they must lie on a line orthogonal to the weight vector

\[ w^T p = ||w|| ||p|| \cos \theta \]

\[ \text{proj. of } p \text{ onto } w = ||p|| \cos \theta = \frac{w^T p}{||w||} \]
An Illustrative Example
Boolean OR

\[ \begin{array}{l}
\{ \mathbf{p}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_1 = 0 \} & \{ \mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t_2 = 1 \} \\
\{ \mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_3 = 1 \} & \{ \mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_4 = 1 \}
\end{array} \]

Given the above input-output pairs (p,t), can you find (manually) the weights of a perceptron to do the job?
Boolean OR Solution

1) Pick an admissible decision boundary

2) Weight vector should be orthogonal to the decision boundary.

\[ \mathbf{w}_1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \]

3) Pick a point on the decision boundary to find the bias.

\[ \mathbf{w}_1^T \mathbf{p} + b = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} + b = 0.25 + b = 0 \implies b = \alpha 0.25 \]
Matrix Form
Multiple-Neuron Perceptron

Weights of one neuron in one row of $W$.

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

$$W = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}$$

$$\mathbf{a}_i = hardlim(n_i) = hardlim(i\mathbf{w}^T \mathbf{p} + b_i)$$
Each neuron will have its own decision boundary.

\[ \mathbf{w}_i^T \mathbf{p} + b_i = 0 \]

A single neuron can classify input vectors into two categories.

An S-neuron perceptron can potentially classify input vectors into \(2^S\) categories.
Perceptron Limitations
Perceptron Limitations

- A single layer perceptron can only learn linearly separable problems.
  - Boolean AND function is linearly separable, whereas Boolean XOR function is not.
AND Network

\[ x_1 \quad W_1 = 0.5 \quad W_2 = 0.5 \quad W_0 = -0.8 \quad X_0 = 1 \]
Perceptron Limitations

Linear Decision Boundary

\[ \mathbf{w}^T \mathbf{p} + b = 0 \]

Linearly Inseparable Problems
Perceptron Limitations

For a linearly not-separable problem:
- Would it help if we use more layers of neurons?
- What could be the learning rule for each neuron?

**Solution:** Multilayer networks and the backpropagation learning algorithm

**Boolean XOR**
• More than one layer of perceptrons (with a hardlimiting activation function) can learn any Boolean function.

• However, a learning algorithm for multi-layer perceptrons has not been developed until much later
  – backpropagation algorithm
  – replacing the hardlimiter in the perceptron with a sigmoid activation function
Summary

• So far we have seen how a single neuron with a threshold activation function separates the input space into two.

• We also talked about how more than one nodes may indicate convex (open or closed) regions

• The next slides = Backpropagation algorithm to learn the weights automatically