Module 12

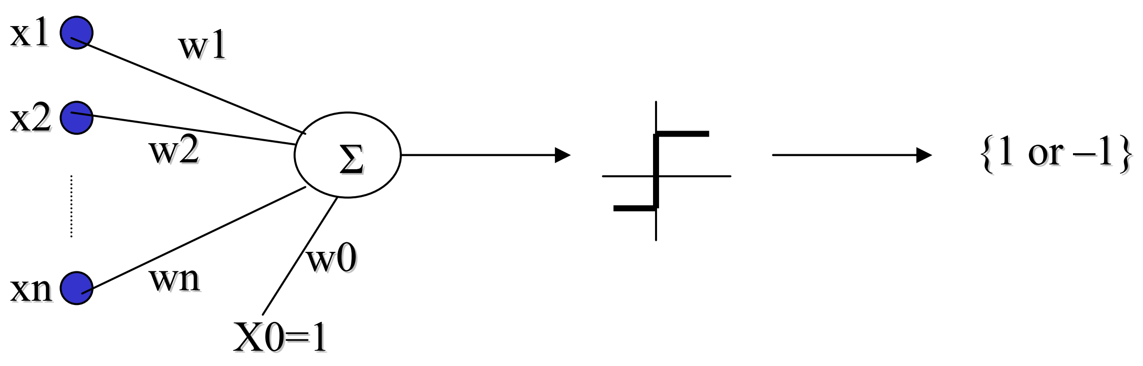
Machine Learning

Lesson 38

Neural Networks - II

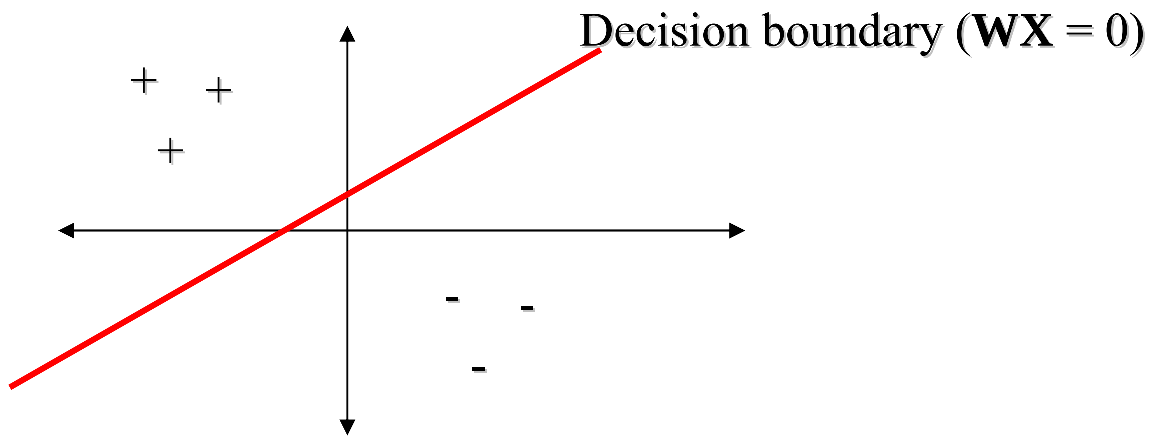
12.4.3 Perceptron

Definition: It's a step function based on a linear combination of real-valued inputs. If the combination is above a threshold it outputs a 1, otherwise it outputs a –1.

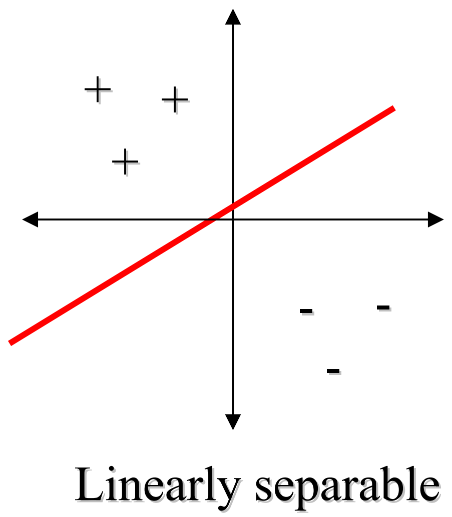


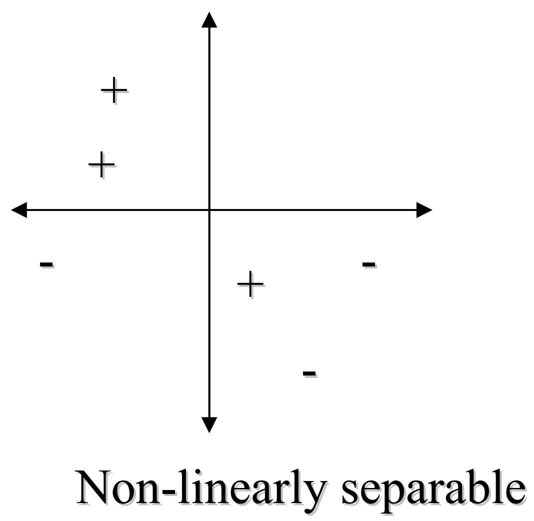
$$O(x1,x2,\ldots,xn) = \begin{cases} 1 \text{ if } w0 + w1x1 + w2x2 + \ldots + wnxn > 0 \\ -1 \text{ otherwise} \end{cases}$$

A perceptron draws a hyperplane as the decision boundary over the (n-dimensional) input space.



A perceptron can learn only examples that are called "linearly separable". These are examples that can be perfectly separated by a hyperplane.

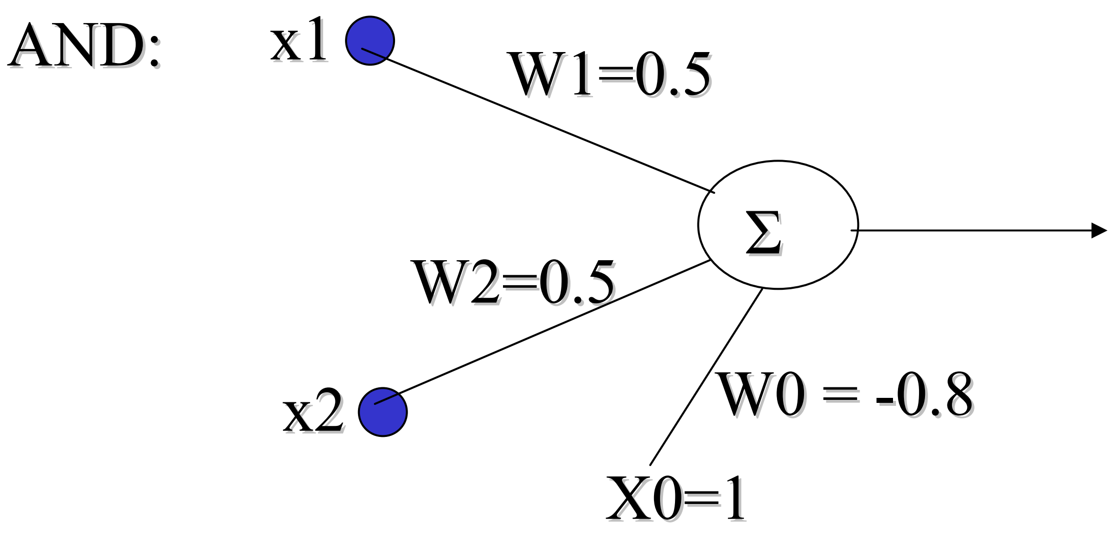




Perceptrons can learn many boolean functions: AND, OR, NAND, NOR, but not XOR

However, every boolean function can be represented with a perceptron network that has two levels of depth or more.

The weights of a perceptron implementing the AND function is shown below.



12.4.3.1 Perceptron Learning

Learning a perceptron means finding the right values for W. The hypothesis space of a perceptron is the space of all weight vectors. The perceptron learning algorithm can be stated as below.

1. Assign random values to the weight vector
2. Apply the *weight update rule* to every training example
3. Are all training examples correctly classified?
    a. Yes. Quit
    b. No. Go back to Step 2.

There are two popular weight update rules.

i) The perceptron rule, and
ii) Delta rule

The Perceptron Rule

For a new training example X = (x1, x2, …, xn), update each weight according to this rule:

$wi = wi + \Delta wi$

Where $\Delta wi = \eta \, (t-o) \, xi$
t: target output
o: output generated by the perceptron
η: constant called the learning rate (e.g., 0.1)
Comments about the perceptron training rule:

- If the example is correctly classified the term (t-o) equals zero, and no update on the weight is necessary.
- If the perceptron outputs –1 and the real answer is 1, the weight is increased.
- If the perceptron outputs a 1 and the real answer is -1, the weight is decreased.
- Provided the examples are linearly separable and a small value for η is used, the rule is proved to classify all training examples correctly (i.e, is consistent with the training data).

The Delta Rule

What happens if the examples are not linearly separable?
To address this situation we try to approximate the real concept using the delta rule.

The key idea is to use a *gradient descent search*. We will try to minimize the following error:

$E = ½ \; \Sigma i \; (ti – oi) \; 2$

where the sum goes over all training examples. Here oi is the inner product WX and not sgn(WX) as with the perceptron rule. The idea is to find a minimum in the space of weights and the error function E.

The delta rule is as follows:

For a new training example X = (x1, x2, …, xn), update each weight according to this rule:

$wi = wi + \Delta wi$

Where $\Delta wi = -\eta\ E'(\mathbf{W})/wi$

η: learning rate (e.g., 0.1)

It is easy to see that

$E'(W)/ wi = \Sigma_i (ti - oi) (-xi)$

So that gives us the following equation:

$wi = \eta\ \Sigma_i (ti - oi)\ xi$

There are two differences between the perceptron and the delta rule. The perceptron is based on an output from a step function, whereas the delta rule uses the linear combination of inputs directly. The perceptron is guaranteed to converge to a consistent hypothesis assuming the data is linearly separable. The delta rules converges in the limit but it does not need the condition of linearly separable data.

There are two main difficulties with the gradient descent method:

1. Convergence to a minimum may take a long time.

2. There is no guarantee we will find the global minimum.

These are handled by using momentum terms and random perturbations to the weight vectors.