Multi-layer Perceptrons

Xiang Zhuoya

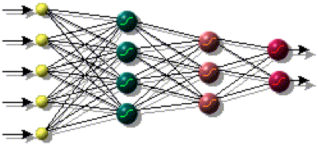
zxiang@cs.ust.hk

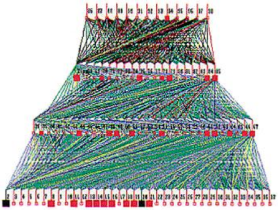
February 24, 2015

Multi-layer Perceptrons

Threshold logic unit (TLU) is also called **perceptron**.

- **Multi-layer perceptrons (MLP)** consist of multiple layers of nodes in a directed graph.
- **MLP** is a modification of the standard linear perceptron and can distinguish data that are **not linearly separable**.





XOR

Consider the logic function XOR, and its truth table is shown below.

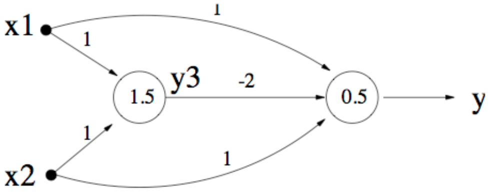
x_1	x_2	$y = \text{XOR}(x_1, x2)$
0	0	0
0	1	1
1	0	1
1	1	0

We can see that XOR is **not linearly separable**.

Solving XOR with a hidden unit

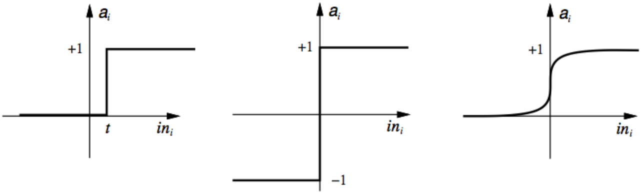
x_1	x_2	y_3	$y = XOR(x_1, x2)$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Adding a hidden unit can solve the XOR problem



Sigmoid Unit

- A unit very much like a perceptron, but based on a smoothed, differentiable threshold function: $\sigma(x) = \frac{1}{1+e^{-x}}$

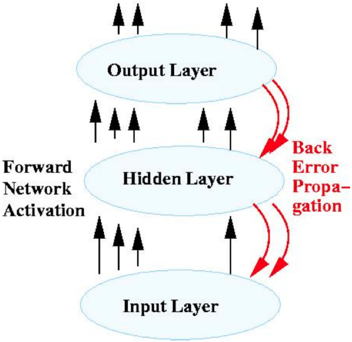


(a) Step function (b) Sign function (c) Sigmoid function

- Nonlinear transfer functions + multi-layer networks requires more sophisticated learning algorithms
- **Back Propagation**

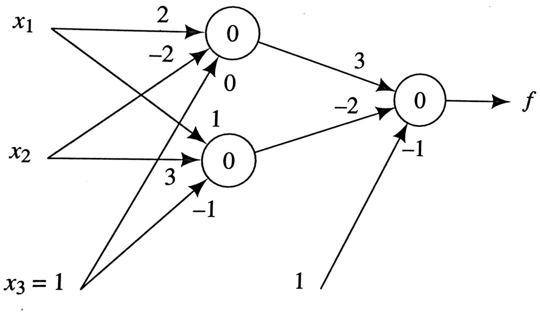
Back Propagation

Idea : Use differentiable transfer functions (e.g. Sigmoid) and differentiable error function.
Goal : Evaluate the derivatives of the error with respect to the weights and update the weights to minimize the error function.



In-class exercise

Exercise on back propagation



Starting with the random weights shown in the picture, our target is to train an even-parity function of two binary variables.

In-class exercise (cont'd)

The inputs and the desired labels are:

x_1	x_2	x_3	d
1	0	1	0
0	0	1	1
0	1	1	0
1	1	1	1

The **backpropagation learning** can be divided into two phases:

Phase 1: Propagation

- Forward propagation of an input to generate the outputs.
- Backward propagation of the output to generate the deltas.

Phase 2: Weight update

- Multiply its output delta and input to get the gradient.
- Subtract a ratio (learning rate) of the gradient from the weight.

Solution

Detailed steps will be shown on the board in class.

For the first input vector $(1, 0, 1)$:

- The first-layer outputs are $f_1 = 0.881, f_2 = 0.500$; and the final output is $f = 0.665$.
- Output delta is calculated as $\delta^{(2)} = -0.148$; backpropagating this δ in the second-layer produces $\delta_1^{(1)} = -0.047$ and $\delta_2^{(2)} = 0.074$.
- With the learning rate $c = 1$, the new weights are calculated to be
  $\mathbf{W}_1^{(1)} = (1.953, -2.000, -0.047)$
  $\mathbf{W}_2^{(1)} = (1.074, 3.000, -0.926)$
  $\mathbf{W}^{(2)} = (2.870, -2.074, -1.148)$

More exercises

More exercises on the tutorial page.