COMP3411 Artificial Intelligence
Session 1, 2016

Tutorial Solutions - Week 11

This page was last updated: 05/17/2016 01:51:52

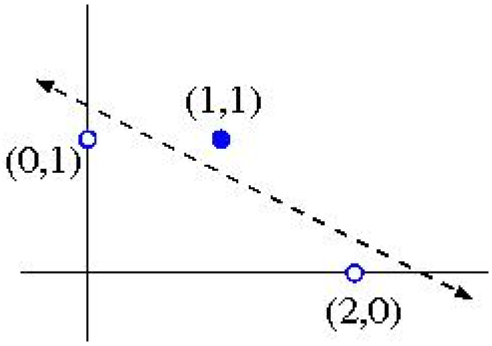
---

1. **Perceptron Learning**

   a. Construct by hand a Perceptron which correctly classifies the following data; use your knowledge of plane geometry to choose appropriate values for the weights w_0, w_1 and w_2.

Training Example	x_1	x_2	Class
a.	0	1	-1
b.	2	0	-1
c.	1	1	+1

The first step is to plot the data on a 2-D graph, and draw a line which separates the positive from the negative data points:



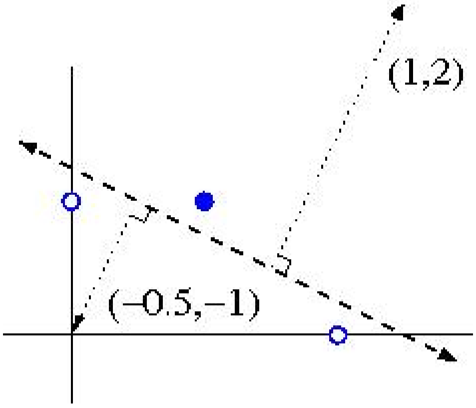
This line has slope -1/2 and x_2-intersect 5/4, so its equation is:

$x_2 = 5/4 - x_1/2$,
i.e. $2x_1 + 4x_2 - 5 = 0$.

Taking account of which side is positive, this corresponds to these weights:

$w_0 = -5$
$w_1 = 2$
$w_2 = 4$

Alternatively, we can derive weights $w_1=1$ and $w_2=2$ by drawing a vector normal to the separating line, in the direction pointing towards the positive data points:



The bias weight w_0 can then be found by computing the dot product of the normal vector with a perpendicular vector from the separating line to the origin. In this case $w_0 = 1(-0.5) + 2(-1) = -2.5$

(Note: these weights differ from the previous ones by a normalizing constant, which is fine for a Perceptron)

b. Demonstrate the Perceptron Learning Algorithm on the above data, using a learning rate of 1.0 and initial weight values of

$w_0 = -0.5$
$w_1 = 0$
$w_2 = 1$

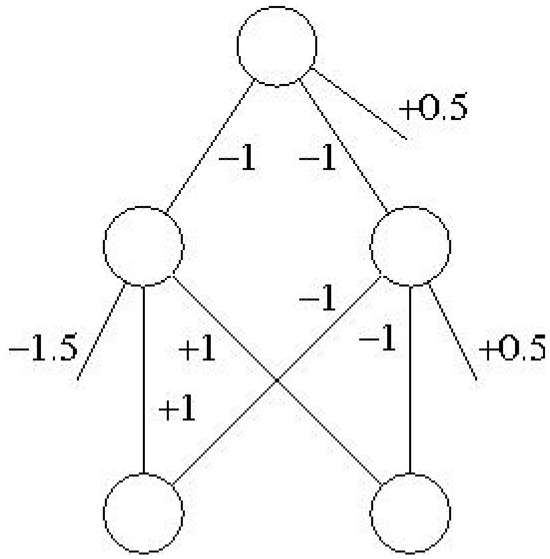
In your answer, you should clearly indicate the new weight values at the end of each training step.

Iteration	w_0	w_1	w_2	Training Example	x_1	x_2	Class	$s=w_0+w_1x_1+w_2x_2$	Action
1	-0.5	0	1	a.	0	1	-	+0.5	Subtract
2	-1.5	0	0	b.	2	0	-	-1.5	None
3	-1.5	0	0	c.	1	1	+	-1.5	Add
4	-0.5	1	1	a.	0	1	-	+0.5	Subtract
5	-1.5	1	0	b.	2	0	-	+0.5	Subtract
6	-2.5	-1	0	c.	1	1	+	-3.5	Add
7	-1.5	0	1	a.	0	1	-	-0.5	None
8	-1.5	0	1	b.	2	0	-	-1.5	None
9	-1.5	0	1	c.	1	1	+	-0.5	Add
10	-0.5	1	2	a.	0	1	-	+1.5	Subtract
11	-1.5	1	1	b.	2	0	-	+0.5	Subtract
12	-2.5	-1	1	c.	1	1	+	-2.5	Add
13	-1.5	0	2	a.	0	1	-	+0.5	Subtract
14	-2.5	0	1	b.	2	0	-	-2.5	None
15	-2.5	0	1	c.	1	1	+	-1.5	Add
16	-1.5	1	2	a.	0	1	-	+0.5	Subtract
17	-2.5	1	1	b.	2	0	-	-0.5	None
18	-2.5	1	1	c.	1	1	+	-0.5	Add
19	-1.5	2	2	a.	0	1	-	+0.5	Subtract
20	-2.5	2	1	b.	2	0	-	+1.5	Subtract

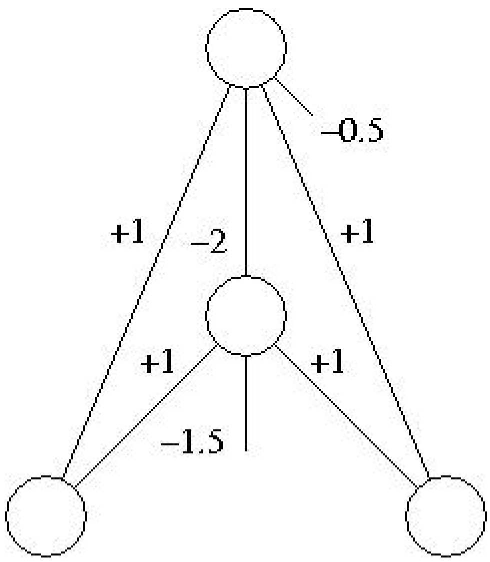
21	-3.5	0	1	c.	1	1	+	-2.5	Add
22	-2.5	1	2	a.	0	1	-	-0.5	None
23	-2.5	1	2	b.	2	0	-	-0.5	None
24	-2.5	1	2	c.	1	1	+	+0.5	None

2. **18.21** Construct by hand a Neural Network (or Multi-Layer Perceptron) that computes the XOR function of two inputs. Make sure the connections, weights and biases of your network are clearly visible.

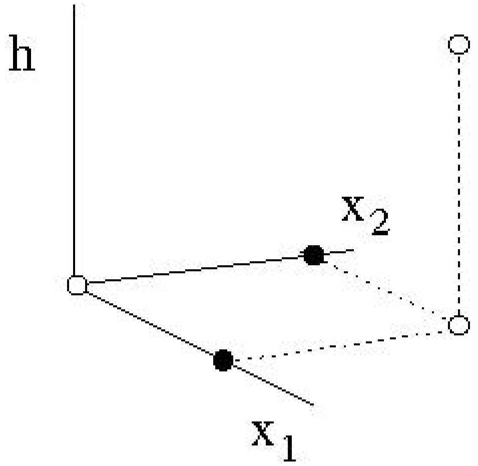
x_1 XOR x_2 can be written as (x_1 AND x_2) NOR (x_1 NOR x_2). This decomposition allows us to compute XOR with a network like this:



Alternatively, XOR can be computed using a network with only one hidden node, provided we also include "shortcut" connections direct from the inputs to the output:



Note that the single hidden unit computes (x_1 AND x_2). The addition of this hidden "feature" creates a 3-dimensional space in which the points can be linearly separated by a plane. The weights for the output unit (+1,+1,-2) specify a vector perpendicular to the separating plane, and its distance from the origin is determined by the output bias divided by the length of this vector.



- Explain how each of the following could be constructed:

  a. Perceptron to compute the OR function of *m* inputs

     Set the bias weight to -½, all other weights to 1.

  b. Perceptron to compute the AND function of *n* inputs

     Set the bias weight to (½ - *n*), all other weights to 1.

  c. 2-Layer Neural Network to compute any (given) logical expression, assuming it is written in Conjunctive Normal Form.

     Each hidden node should compute one disjunctive term in the expression. The weights should be -1 for items that are negated, +1 for the others. The bias should be (*k* - ½) where *k* is the number of items that are negated. The output node then computes the conjunction of all the hidden nodes, as in part b.
