COMP4670/6467
Introduction to Statistical Machine Learning
Tutorial 4

Christfried Webers

26/28 March 2013

1 The XOR function

The XOR function maps a two-dimensional input from a binary domain (e.g. $\mathbf{x} \in \{0,1\}^2$) into a one-dimensional output (e.g. $\text{XOR}(x_1, x_2) \in \{0,1\}$) in the following way

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

Table 1: Definition of the function $\text{XOR} : \{0,1\} \times \{0,1\} \to \{0,1\}$.

2 Neural Network for XOR function

Implement a neural network with one hidden layer to approximate the XOR function as a mapping $\text{XOR} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. In each layer where a bias is feasible, allow it to be switched on or off. Compare the results for the two different nonlinear activation functions $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$, and $\sigma(a) = \frac{1}{1+e^{-a}}$.

3 Visualisation

Given a trained network for the XOR function, the solution can be visualised by using the following code where *Calc_Output* uses the trained network to map the inputs into outputs.

```
import numpy
import pylab
...
delta = 0.01
x     = numpy.arange(-1.0, 2.0, delta)
y     = numpy.arange(-1.0, 2.0, delta)
X, Y = numpy.meshgrid(x, y)

Z = numpy.empty(X.shape)
for i in xrange(Z.shape[0]):
    for j in xrange(Z.shape[1]):
        input = [X[i,j], Y[i,j]]
        Z[i,j] = Calc_Output(input)

number_of_contours = 10
plot = pylab.contour(X, Y, Z, number_of_contours)
pylab.clabel(plot, fontsize=9)

# Draw the four points
pylab.plot([0, 1], [0, 1], 'bo')
pylab.plot([0, 1], [1, 0], 'ro')
pylab.show()
```

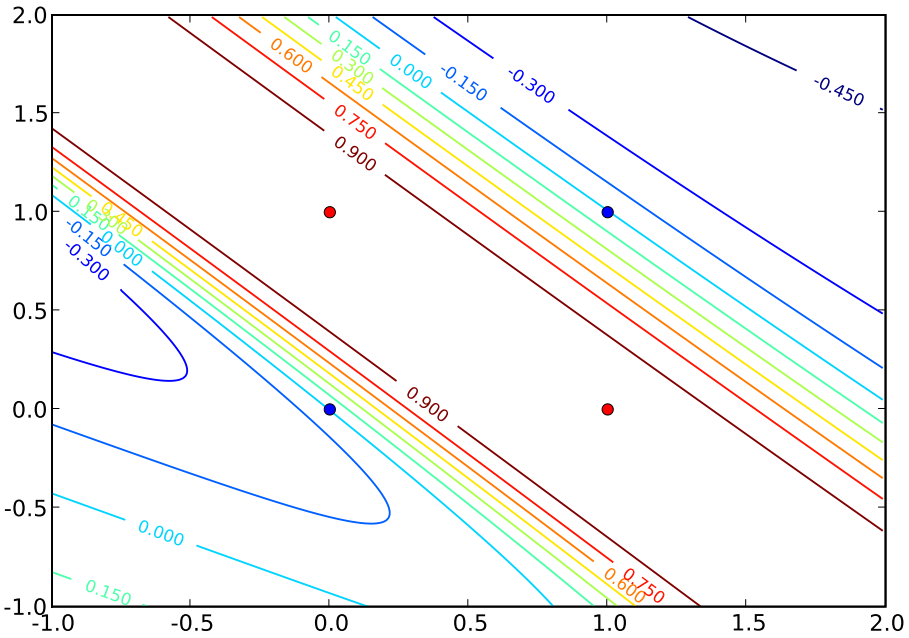

Figure 1: Contour plot for a two-layer network approximating the XOR function with three hidden nodes and the tanh(·) activation function. (No bias in the hidden layer.)