Exercise 8 - Machine Learning II - 2016

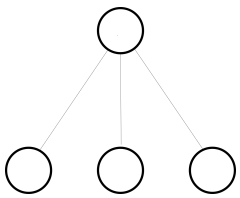
Please send your submissions (runnable code, plots and written answers) **via email to weis@ccc.cs.uni-frankfurt.de until Tuesday June 14th 2016**. One submission per student. Prepare to present your solutions in the exercise session. Students that are not able to explain their solutions may not be given credit on their submissions.

1 Neural Networks implementation (5 Points)

In this exercise you will implement and train a small Neural Network with backpropagation. Consider the following data:

Input			Output
0	0	1	0
1	1	1	1
1	0	1	1
0	1	1	0

The goal is to predict the output, given the three inputs, using the following network architecture:



Lets use the following definitions and variables in this exercise:

Variable	Definition
X	Input dataset matrix (each row is a training example)
y	Output dataset matrix (each row is a traning example)
l0	First layer of the net (specified by input data)
l1	Second layer of the net (usually called hidden, here:output)
w0	Weights of the first layer, connection first to second layer
*	Elementwise multiplication
-	Elementwise subtraction
x.dot(y)	Dot product

As you already know, you need to implement the following steps:

1. Implement a function that returns the logistic (hint: $\frac{1}{1+e^{-x}}$) as nonlinearity, as well as it's derivative given x (hint: $logistic(x) * (1 - logistic(x))$)
2. Represent input and output dataset as numpy arrays
3. Initialize the weights randomly as numpy array (use np.random.seed(1) beforehand to get a deterministic random sequence for comparison reasons)
4. In a loop (1000 times):
   - Populate the first layer with the training data (all at once, full batch)
   - Compute the forward-pass (prediction) (hint: logistic(np.dot()))
   - Compute the error w.r.t. the given outputs
   - Perform the weight-update (hint: multiply error with derivative of predicted values)
   - Visualize the current error (np.sum(np.abs(error)))

2 Neural Networks - Questions (5 Points)

- Why do we pass the output of $l0 * w0$ through the nonlinearity?
- What is the meaning of weighting the error with the derivative of the output prediction (what happens for small/large errors)?
- How could you automatically stop the training process without specifying a constant number of iterations?
- As you noticed, the first variable is directly correlated with the output, we have a linear mapping of input to output. How can the network structure be changed to handle nonlinear problems?
- What are the differences of Stochastic Gradient Descent to the method used in our example?