Submission until: **14.06.2015**
Discussion on: 16.06.2015
Submission as upload to your groups stud.IP folder as *groupNumber_sheet7.zip*

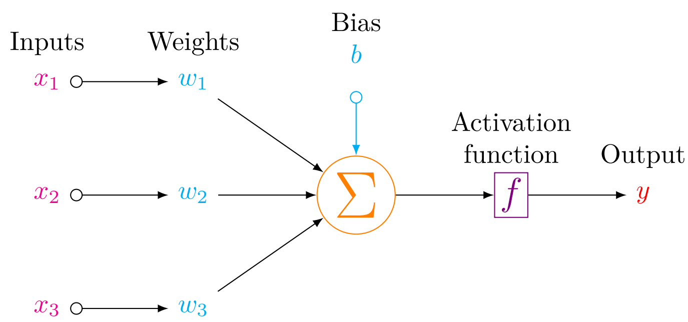

Figure 1: Neural network – single neuron perceptron

Assignment 1 *(Perceptron (6p))*

1. Sketch a single neuron perceptron with weights and threshold θ for the following logical functions:
   (a) $(A \wedge B) \vee (\neg A \wedge B)$
   (b) $(A \wedge B) \vee (\neg A \wedge B) \vee (A \wedge \neg B)$
2. Draw pictures similar to slide page 36 in chapter 7 for the boolean functions below, and give weights for the perceptron or argue if no such weight exist. The features x_1, x_2 and output y can only take values -1 or +1.
   (a) $y = \begin{cases} +1 & if\ x_1 = 1\ and\ x_2 = -1 \\ -1 & \text{otherwise} \end{cases}$

Assignment 2 *(Perceptron (6p))*

Simple classification problem for single neuron perceptron.

1. Draw a decision boundary in figure 2 (a) and (b) and find weight and bias values for each network (Hint: find the bias values for each perceptron by picking a point on the decision boundary)
2. Determine the margin of the the separation line for each figure. (Hint: Compute the distance between each point in the figure to the selected decision boundary)

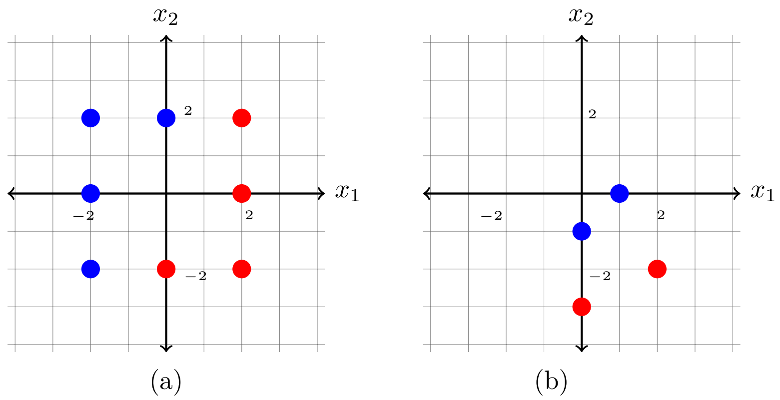

Figure 2: Simple classification problems. Blue dot represents positive and red dot for negative.

Assignment 3 *(Perceptron (10p))*

Algorithm:

1. Initialize weights and biases randomly
2. While stop condition is false:

   For each training pair (x, t):

   2.1. Compute $y_{in} = \Sigma(x \cdot w)$

   2.2. $y = \begin{cases} 1 & \text{if} \quad y_{in} > \theta \\ 0 & \text{if} \quad -\theta \leq y_{in} \leq \theta \\ -1 & \text{if} \quad y_{in} < -\theta \end{cases}$

   2.3. Update weights (and biases [1]) using incremental mode mentioned in the lecture:
   $\triangle w = \epsilon \cdot (t - y) \cdot x$ where t is a target and ϵ is a learning rate
   $w_{new} = w_{old} + \triangle w$

3. Repeat until convergence i.e. no error from output

Tasks:

1. Train noisy logical OR signals, "train_or.mat[2]" using perceptron with incremental mode to update the weight. Plot the input units using the same scale on axes. Apply different colors for positive and negative outputs. Draw a decision line. (6p) Test data from "test_or.mat[3]" and plot the outputs with colors (or markers) to easily identify the output types on the same plot from training inputs. (2p)
2. Now, apply the code to train standardized *oldfaithful* data, "train_oldfaithful.mat[4]". Use the trained weights to test "test_oldfaithful.mat[5]" and plot the data on the same plot as the training data with different markers (or colors) corresponding to output types. (2p)

[1] To update bias, you may use $w0_{new} = w0_{old} + \epsilon \cdot (t - y)$
[2] Training input parameter is marked as "x" and target output as "t". Python user can use keys() from returning of scipy.io.loadmat(filename) in order to list the keys in dict which saved in mat file.
[3] Testing input parameter is marked as "x_test"
[4] Training input parameter is marked as "x" and target output as "t"
[5] Testing input parameter is marked as "x_test"