Beihang University
School of General Engineering

Beijing, 8th, June
Lin Wei and Yida Ding

Data Management and Artificial Intelligence
Lab Class 14

Task 1 *Perceptron*

Please apply the algorithm in your homework to a more general case.

1. As shown in Figure 25, implement a function which generate n random points in $[0,1]\times[0,1]$ and separate them with line $2x+3y-2=0$. Using matplotlib to visualize the results.

2. Please implement a single perceptron in Python that is able to give an output with specific input values. The algorithm should start with randomly generated weights $\mathbf{w}$. It should be able to find the appropriate values for the weights by updating it iteratively with a given supervised learning method. Then test your code with different activation functions and variables of α, i.e., the coefficient for updating the perceptron weights w.

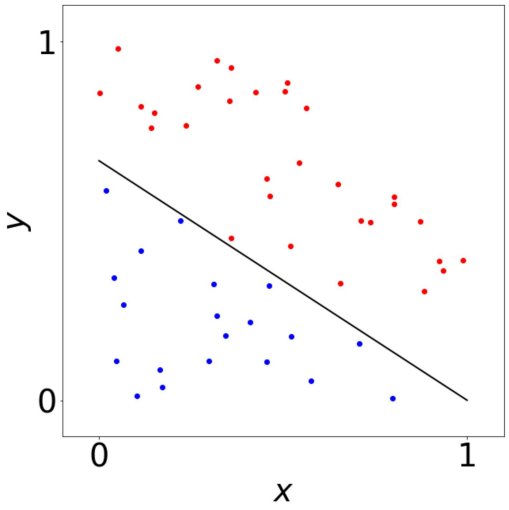

Figure 25: Point separation by line $2x+3y-2=0$

Task 2 *Neural network*

Please implement a Python algorithm for a small neural network with **nonlinear** decision boundaries. Just following the concept of stacking up neurons in the lecture. Using the operation "XOR" as an instance, you should implement the following sub-tasks:

1. Implement a function for the forward propagation process. For each pair of input (x_0, x_1) and a given list of weight matrices, it should output a value $y \in \{0,1\}$.

2. Try to find the appropriate values of the weight matrices. With these values, the above function will implement the XOR operation.

Hint: $XOR(x_0, x_1) = OR(x_0, x_1) \wedge \neg AND(x_0, x_1)$. This equation will help you find the appropriate weights.

Task 3 ***Logistic regression***

If the data are not linearly separable, then the algorithm in task1 will not converge. Try to deal with the situation that some red points and blue points are mixed together. Our goal is to minimize the mean-square-error (MSE). Let $\vec{X} = \{\vec{x_j}\}_{j=1}^{N}$ and $\vec{Y} = \{\vec{y_j}\}_{j=1}^{N}$ be the coordinates and class labels of the points in the training data. The MSE is computed as follow:

$$MSE(\vec{w}, \vec{X}, \vec{Y}) = \sum_{j=1}^{N} (y_j - g(s(\vec{x_j}, \vec{w_j})))^2$$

where $g(s) = \frac{1}{1+e^{-s}}$ is the activation function and $s(\vec{x_j}, \vec{w_j}) = \vec{x_j}^T \vec{w_j}$. And the update formulation for each w_i in dimension i can be as follows:

$$w_i^{r+1} = w_i^r + 2\alpha \left[\sum_{j=1}^{N} (y_j - g(s(\vec{x_j}, \vec{w_j}))) * g(s(\vec{x_j}, \vec{w_j})) * (1 - g(s(\vec{x_j}, \vec{w_j})))x_{ji} \right]$$

The desired output is shown on Fig 26.

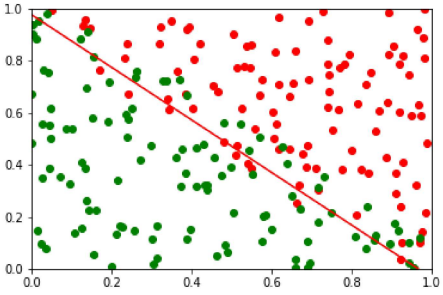

Figure 26: Logistic regression