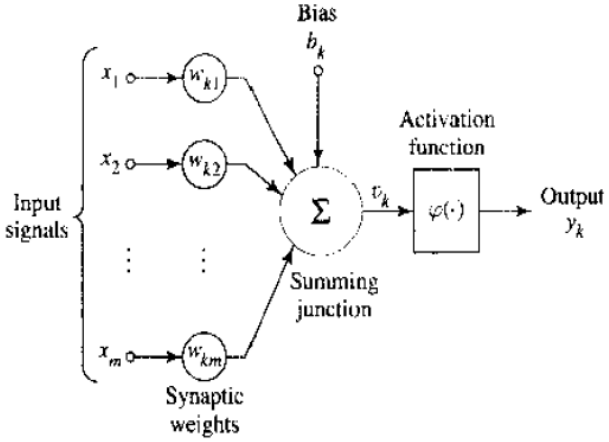
# EE5904/ME5404 Neural Networks: Homework #1

**Important note:** the due date is 08/02/2021. You should submit your scripts to the folder in LumiNus. Late submission is not allowed unless it is well justified. Please include the MATLAB code or Python Code as attachment if computer experiment is involved.

**Q1**. (10 Marks)




Consider the signal-flow graph of the perceptron shown in the above figure. The activation function, $\varphi(v)$, where v is the induced local field, can be designed by the user. If the activation function is chosen as hard limiter (i.e. step function), then it becomes the classical perceptron, and the decision boundary is shown to be a hyperplane. In this problem, let's explore other choices of the activation function, and its effect on the decision boundary. Let's assume that the classification decision made by the perceptron is simply a threshold rule defined as follows:

*Observation vector* $x = [x_1 \; x_2 \; \cdots \; x_m \;]^T$ *belongs to class* $C_1$ *if the output* $y > \xi$, *where* $\xi$ *is a user-defined threshold; otherwise, x belongs to class* $C_2$.

Consider the following three choices of activation function:

1) The activation function is a quadratic function: $\varphi(v) = (v-a)^2 + \mathrm{c}$;

2) The activation function is the hyperbolic tangent function: $\varphi(v) = \dfrac{1-e^{-v}}{1+e^{-v}}$;

3) The activation function is the Bell-shaped Gaussian function: $\varphi(v) = e^{-\frac{(v-m)^2}{2}}$.

For each case, investigate whether the resulting decision boundary is a hyper-plane or not.

**Q2**. (10 Marks)
Consider the logic function, EXCLUSIVE OR (XOR).

Truth Table of XOR

| | | | | |
|---|---|---|---|---|
| $x_1$ | 0 | 1 | 0 | 1 |
| $x_2$ | 0 | 0 | 1 | 1 |
| $y$ | 0 | 1 | 1 | 0 |

It is well known that the XOR problem is not linearly separable. It seems obvious by visually checking, which however cannot be accepted as mathematical proof. Therefore, please supply a rigorous mathematical proof for this statement.

**Q3**. (10 Marks)
The perceptron could be used to perform numerous logic functions, such as AND, OR, COMPLEMENT and NAND function, whose truth tables are tabulated as follows respectively.

| | | | | |
|---|---|---|---|---|
| x1 | 0 | 0 | 1 | 1 |
| x2 | 0 | 1 | 0 | 1 |
| y | 0 | 0 | 0 | 1 |

AND

| | | | | |
|---|---|---|---|---|
| x1 | 0 | 0 | 1 | 1 |
| x2 | 0 | 1 | 0 | 1 |
| y | 0 | 1 | 1 | 1 |

OR

| | | |
|---|---|---|
| x | 0 | 1 |
| y | 1 | 0 |

COMPLEMENT

| | | | | |
|---|---|---|---|---|
| x1 | 0 | 0 | 1 | 1 |
| x2 | 0 | 1 | 0 | 1 |
| y | 1 | 1 | 1 | 0 |

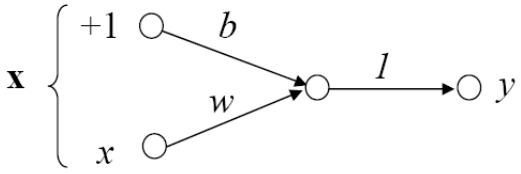
NAND

a). Demonstrate the implementation of the logic functions AND, OR, COMPLEMENT and NAND with selection of weights by off-line calculations.
(3 Marks)

b). Demonstrate the implementation of the logic functions AND, OR, COMPLEMENT and NAND with selection of weights by learning procedure. Suppose initial weights are chosen randomly and learning rate $\eta$ is 1. Plot out the trajectories of the weights for each case. Compare the results with those obtained in (a). Try other learning rates, and report your observations with different learning rates.
(4 Marks)

c). What would happen if the perceptron is applied to implement the EXCLUSIVE OR function with selection of weights by learning procedure? Suppose initial weight is chosen randomly and learning rate $\eta$ is 1.0. Do the computer experiment and explain your finding. (3 Marks)

**Q4.** (10 Marks)
Single layer perceptron with pure linear activation function can be used to fit a linear model to a set of input-output pairs. Suppose that we are given the following pairs:
{(0.5,8.0), (1.5, 6.0), (3, 5), (4.0, 2), (5.0, 0.5)}
and a single linear neuron as shown in the following figure.




a). Find the solution of w and b using the standard linear least-squares (LLS) method. Plot out the fitting result.

(3 Marks)

b). Suppose that initial weight is chosen randomly and learning rate $\eta$ is 0.02. Find the solution of w and b using the least-mean-square (LMS) algorithm for 100 epochs. Plot out the fitting result and the trajectories of the weights versus learning steps. Will the weights converge?

(3 Marks)

c). Compare the results obtained by LLS and the LMS methods.

(2 Marks)

d) Repeat the simulation study in b) with different learning rates $\eta$, and explain your findings.

(2 Marks)

**Q5.** (10 Marks)
In a variant of the LMS algorithm called the *leaky LMS algorithm*, the cost function to be minimized is defined by

$$E(n) = \frac{1}{2}e^2(n) + \frac{1}{2}\lambda \| w(n) \|^2$$

where w(n) is the weight vector, e(n) is the estimation error, and $\lambda$ is a positive constant. As in the ordinary LMS algorithm, we have the estimation error,

$$e(n) = d(n) - w^T(n)x(n)$$

where d(n) is the desired response corresponding to the input vector x(n).

Following the similar procedure to derive the learning algorithm for LMS, show that the time update for the weight vector of the leaky LMS algorithm is defined by

$$w(n+1) = (1 - \eta\lambda)w(n) + \eta x(n)e(n)$$

which includes the ordinary LMS algorithm as a special case.