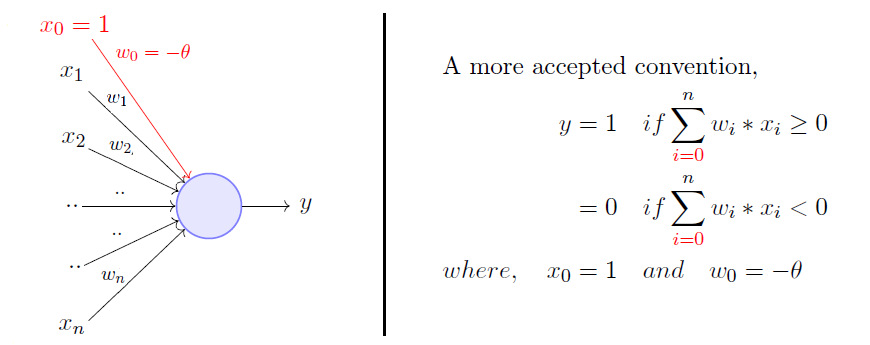
**Perceptron Algorithm for AND, OR, NAND and NOR**

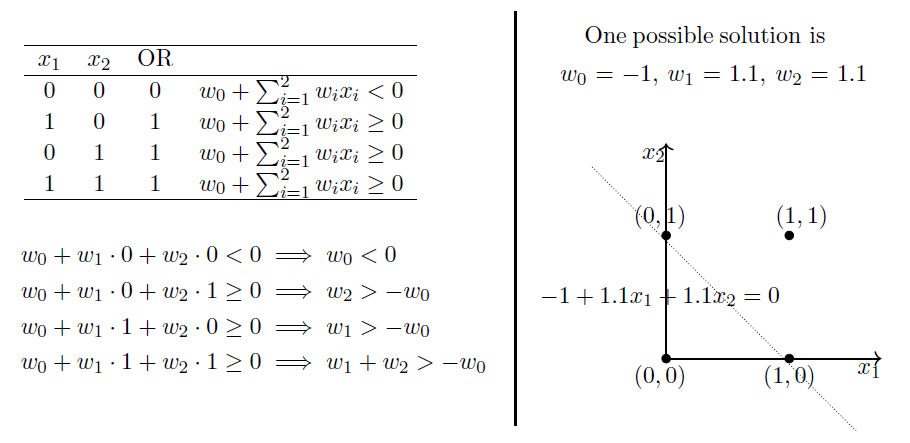
**Vishal Kumar, MIT2019090**

**Perceptron Algorithm:**

The perceptron model takes an input, aggregates it (weighted sum) and returns 1 only if the aggregated sum is more than some threshold else returns 0. Rewriting the threshold as shown above and making it a constant input with a variable weight, we would end up with something like the following:



**OR function using a perceptron:**



Same with AND, NAND and NOR.

Training Data: [[1, 0, 0], [1, 0, 1], [1, 1, 0], [1, 1, 1]] here, x[0] is bias term’s coefficient.

I am taking Initial weights randomly using:

w0 = np.random.randn()

w1 = np.random.randn()

w2 = np.random.randn()

**Observations:**

**For testing my model, I am specifying some testing data:**

test\_data = [[0.98, 1],[0.01, 0.97],[0.77, 0.99],[0.912, 1.002],[0.88, 0.11],[0.82, 0.9],[0.8, 1],[0.02, 0.01],[0.21, 0.99],[0.11, 0.2],[0.79, 1],[0.11, 1.02],[0.98, 0.87],[0.2, 1.3],[0.2, 0.003]]

and Actual outputs for the same is:

test\_op = [1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0] # For AND

test\_op = [1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0] # For OR

test\_op = [0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1] # For NOR

test\_op = [0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1] # For NAND

**For alpha = 0.1, epochs =1000:**

Initial weights -

w0 = -1.3180345632728732 w1 = 1.4075124085250568 w2 = -0.2529847047293955

Final weights -

w0 = 0.4819654367271267 w1 = -0.3924875914749431 w2 = -0.2529847047293955

And Max Accuracy is 100%.