Machine Learning

Assignment 6.1

Submitted By: Ranji Raj

December 4, 2020

Representing a Perceptron

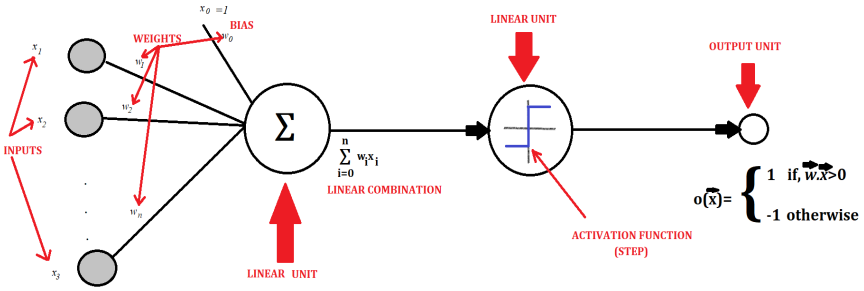

Figure 1: Perceptron with thresholded output

- The Perceptron is the most simplest type of single-layer artificial neural network.
- Perceptrons are basically used for supervised *classification* problems.
- It is the task of learning *thresholded output*.
- It takes a set of input vectors $\vec{x_i}$ that are nothing but the training data.
- Along with these inputs there are set of weights associated which are further learned in the process.
- Bias which corresponds to intercept in linear models are useful for generating more functions with the same set of inputs. It has a special value which is equal to 1.

- The **linear unit** is equivalent to the **simple linear regression** in calculating a value as,

$$\hat{y} = \sum_{i=0}^{n} w_i x_i = w_0 * x_0 + w_1 * x_1 + ... + w_n * x_n$$

- The next part is again a linear unit which is contained by an **activation function** which is basically for classification (commonly used: Binary step function).
- The final **output unit** is used for **classifying** the records as a resultant of the underlying linear transformations through the activation function.
- The above linear computation is processed to a binary step function. For a binary classification problem, if the $\hat{y}$ is > 0 then the unit fires a 1, otherwise, it fires -1.
- The output can also be calculated as, $o(\vec{x}) = \text{sgn}(\vec{w}.\vec{x})$, where sgn is the signum function which decides the sign of the output.

Classification by a Perceptron

- Classification possible only for **linearly separable** problems.
- Ex. Consider the classical example of representing AND-gate and OR-gate.

x1	x2	t
0	0	0
0	1	0
1	0	0
1	1	1

Table 1: Input-Output patterns-AND

x1	x2	t
0	0	0
0	1	1
1	0	1
1	1	1

Table 2: Input-Output patterns-OR

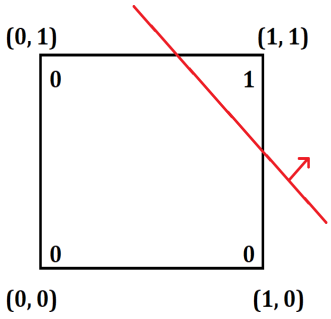

Figure 2: Geometric representation-AND

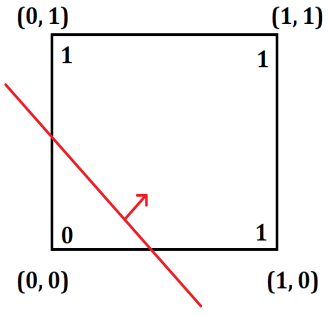

Figure 3: Geometric representation-OR

Now for representing the XOR function a linear separation on a 2D space is not possible.
In order for a perceptron to solve this problem, the following four inequalities must be satisfied:

$$0 * w_1 + 0 * w_2 < \theta = 0 < \theta$$

$$0 * w_1 + 1 * w_2 > \theta = w_1 > \theta$$

$$1 * w_1 + 0 * w_2 > \theta = w_2 > \theta$$

$$1 * w_1 + 1 * w_2 < \theta = w_1 + w_2 < \theta$$

Obviously, it is impossible to have both w_1 and w_2 greater than θ while their sum, w_1 and w_2 is less than θ.

x1	x2	t
0	0	0
0	1	1
1	0	1
1	1	0

Table 3: Input-Output patterns-XOR

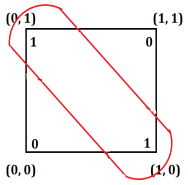

Table 4: Geometric representation-XOR

Learning a Perceptron

- Acceptable weight vector is learned by primarily setting with random weights, then iteratively apply the perceptron to train each training example, modifying the perceptron weights whenever it misclassifies an example.
- The output (o) produced by the perceptron is subtracted from the target (t) and is squared for all the points and the error is recorded. i.e.

$$E[\vec{w}] = \frac{1}{2}\sum(t - o)^2$$

- Until the error minimizes, the weights are refreshed in iterations which is the learning process.
- Weights are modified at each step according to the *perceptron training rule*, which revises the weight w_i, associated with input x_i, according to the rule,

$$w_i \leftarrow w_i + \Delta w_i$$

where,

$$\Delta w_i = \eta(t - o)x_i$$

here, η is the learning rate, a positive constant made to decay as the number of weight-tuning iterations increases.