# Perceptron:

Perceptron is a single layer neural network and a multi-layer perceptron is called Neural Networks.

A (single layer) perceptron is a single layer neural network that works as a linear binary classifier. Being a single layer neural network it can be trained without the use of more advanced algorithms like back propagation and instead can be trained by "stepping towards" your error in steps specified by a learning rate. When someone says perceptron, I usually think of the single layer version.

If you're talking about a multilayer perceptron, however, then the term is the same as a feed-forward neural network.

# Activation Functions:

* Sigmoid Function / Logistic Function (Outputs betn 0 and 1)
* Hyperbolic Tangent (Looks similar to sigmoid. It outputs betn -1 and 1)
* Rectified Linear Unit (ReLU) : If output value is less than 0 we treat it as 0, otherwise we output the actual z value ( ie: max(0,z) ). Has very good performance, especially when dealing with vanishing gradient. Overall very good default.
* Wiki has a list of activation functions that you can look at.

# Multi-Class Classification:

Two main types of multi-class situation:

Non-Exclusive Classes: Where a data point can have multiple classes/categories assigned to it. Ex: Photos can have multiple tags (e.g. beach, family, vacation, etc.. )

Mutually Exclusive Classes (More Common): Only one class per data point. Ex: Photos can be categorized as grayscale or color photo. But cannot be both at the same time.

For classification using neural network: One hot encoding. Means the output layer will have as many nodes as the number of possible labels.

For Non-Exclusive Classes: We can use sigmoid function and we set a cut-off value, for example, say 0.5. For a data point, if an output node's value is >= 0.5 then the data point would be classified by that output node's class. Here a data point can be classified as multiple classes.

For Mutually-Exclusive Classes: We can use softmax function. Here, the function will calculate probabilities of each target class over all possible target classes. But note that sum of all the probabilities will equal to 1. So we will classify a data point as a target class if its output node's probability is maximum among all.

# Cost Function and Gradient Descent:

For regression problem cost function will be quadratic cost function which looks like mean square error. (Similar to linear regression cost function in Andrew NG)

For classification problem, we‘ll use cross entropy loss function (Similar to logistic regression cost function in Andrew NG)

For gradient descent, we’ll choose adam optimizer. It performs best among all gradient descents. Basically, if we choose a random large step size adam optimizer automatically adapts it so that we don’t overshoot and undershoot the global minima and also converge to our solution efficiently.