Perceptron:

         Invented in 1957 at the [Cornell Aeronautical Laboratory](http://en.wikipedia.org/wiki/Cornell_Aeronautical_Laboratory) by [Frank Rosenblatt](http://en.wikipedia.org/wiki/Frank_Rosenblatt).

         Funded by the United States [Office of Naval Research](http://en.wikipedia.org/wiki/Office_of_Naval_Research). Used to distinguish tanks from their surrounding environment.

         This machine was designed for image recognition: it had an array of 400 [photocells](http://en.wikipedia.org/wiki/Photocell), randomly connected to the "neurons". Weights were encoded in [potentiometers](http://en.wikipedia.org/wiki/Potentiometer), and weight updates during learning were performed by electric motors.

         It was later realized that the perceptron was influenced not only by the shapes of images given for its interpretation but it also was effected by the brightness and was unable to clarify the presence of tanks when there was a different brightness to the time the tank present data was taken.

How does the perceptron work?

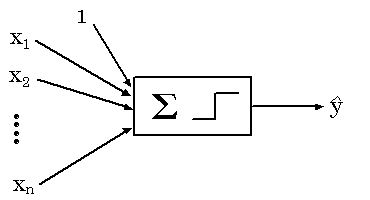


Figure 1. :  is a graphical illustration of a perceptron with inputs*http://reference.wolfram.com/applications/neuralnetworks/NeuralNetworkTheory/NBMLImages/2.4.0/2.4.0_1.gif, ..., http://reference.wolfram.com/applications/neuralnetworks/NeuralNetworkTheory/NBMLImages/2.4.0/2.4.0_2.gif*and output http://reference.wolfram.com/applications/neuralnetworks/NeuralNetworkTheory/NBMLImages/2.4.0/2.4.0_3.gif (sourced from <http://reference.wolfram.com/applications/neuralnetworks/NeuralNetworkTheory/2.4.0.html>)

As seen in figure 1 the weighted sum of the inputs and the unity bias are first summed and followed by being processed by a step function yielding the output

* (x, w, b)= UnitStep (w1 x1 + w­2 x­2 + . . . + wn xn + b)

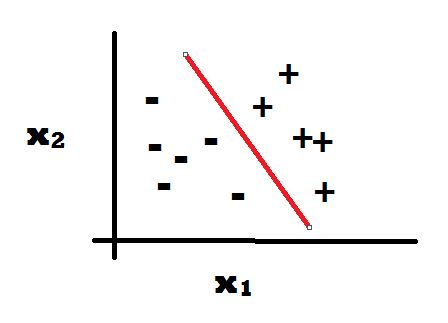
Where {w1. . . wn} are the weights applied to the input vector and b is the bias weight. Each of the weights are represented by the arrows in figure 1. The Unitstep function is 0 for arguments less than 0 and 1 elsewhere. So http://reference.wolfram.com/applications/neuralnetworks/NeuralNetworkTheory/NBMLImages/2.4.0/2.4.0_3.gif can take values of 0 or 1 depending on the value of the weighted sum. The perceptron can indicate 2 classes corresponding to these 2 input values. While in the training process, the weights (inputs and bias) are adjusted so the input data is mapped correctly to one of the two classes.

**Off sample performance more important!!!**

**Cross validation**

Cross-validation, sometimes called rotation estimation, is a [model validation](http://en.wikipedia.org/wiki/Model_validation) technique for assessing how the results of a [statistical](http://en.wikipedia.org/wiki/Statistics) analysis will generalize to an independent data set. Generally better for a scenario where the goals are predicted, and the test is to see how accurate the prediction is by having a training set and a validation set. The goal of cross validation is to define a dataset to "test" the model in the training phase (i.e., the *validation dataset*), in order to limit problems like overfitting, give an insight on how the model will generalize to an independent data set (i.e., an unknown dataset, for instance from a real problem).

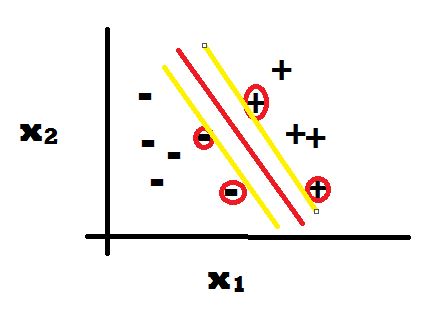
A bigger data set presumably gives a better generalization!

figure 1: the line between +’s and –‘s is constantly moving until it fits to a particular position where the margin between the –‘s and +’s are equal.

A problem with this will be if they aren’t separated appropriately. 3 common problems causing this may be outliers, noise and mislabelled points. It is best to keep the margin between +’s and –‘s as large as possible.

Margin= distance from separating surface to nearest point, assuming points are correct.

The closest points to the separating surface are known as support vectors and they are used to help calculate the maximum margin.

Figure 2: the +’s and –‘s with circles around them are support vectors. The distance between the red and yellow lines is the margin.