**LINEAR CLASSFICATION**

Linear Classification algorithms have no way of optimising the the space between labels.

What we want is to maximise the margin between positive and negative labels.

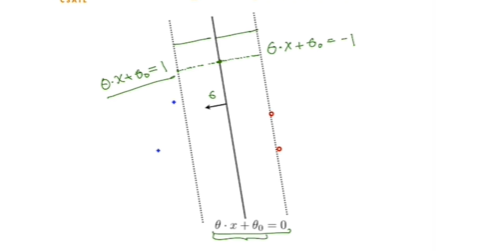
We want a large margin classifier.

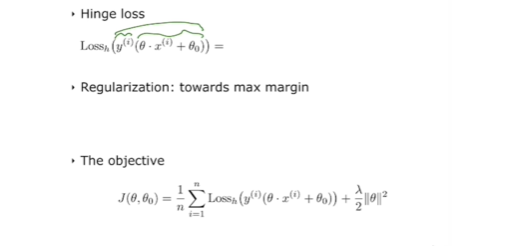
We want a central linear classifier and equidistant large margin boundaries on either side and the negative and positive labels outside of those margin boundaries

Therefore, we have and objective function that has includes regularisation pushing the equidistant boundaries plus a loss function because the loss increase if the labels are within the margin boundaries.

Objective function = loss + regularisation

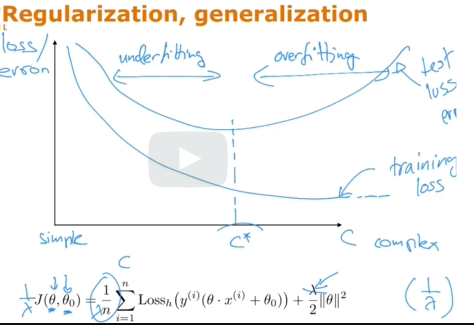
We want to maximise regularisation (push out the boundaires) and minimise the loss.





Minimise after Lossh. Multiplying the linear classifier line by the label value. If in agreement (value above 1) then no loss but if the agreement is less than 1 then their is loss. The lamda/2 is the regularisation term. Minimising this, pushes the boundaries apart. The larger value of lambda is pushing the boundaries apart, the smaller value of lambda, the more emphasis is on reducing the loss.

Large lamdba lets in misclassified points.



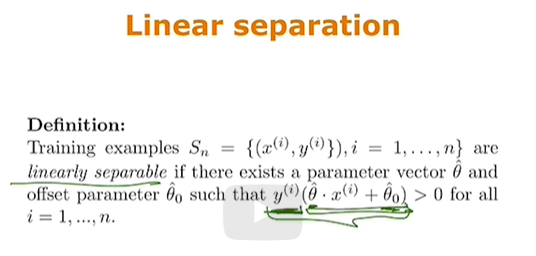
Here, we divide the whole equation by lambda and call 1/lambda = C

As C increases, we emphasis minimisaton of losses and when C is small, we emphasize simpler maximisation of boundaries so more losses.

The plot is of theta and theta zero for various of values of c.

The best generalising method is when C = C\* cause that miniises the test loss.

**LINEAR CLASSIFICTION ALGORITHMS**



Correct classifications happen when;

Label is -1 &

-1\* theta\*feature vector + theta\_zero is positive

Label is +1 &

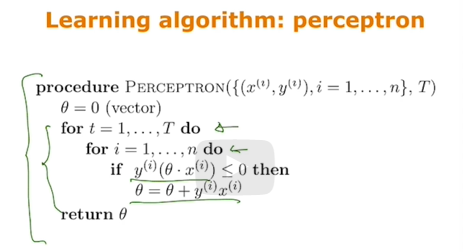
+1\* theta\*feature vector + theta\_zero is positive

Where theta\_zero is the offset paramter or the line would be through the origin

Training error is number of missclassfied.

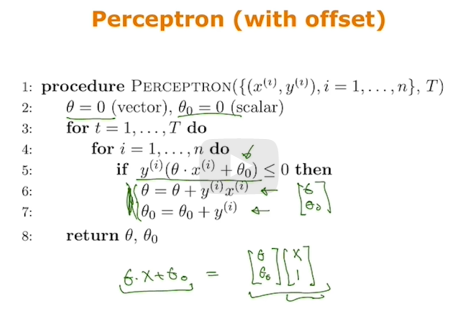
**Perceptron Algorithm**

Finds a linear classifier through the origin that correctly classified all points if such a classifier exists.



If negative or equal (incorrectly classified) then change theta

Keep repeat as earlier updates maybe become misclassified by later updates.



Adding the offset so it is not only searching for a classifier through the origin.