Training of Iris flower dataset and comparing performance of different Algorithms

Author – Kornia Bapari

Sections –

1 . Libraries used in the code

2. Dataset used in the code

3. Dataset walk Through

4. Dataset through Graphs

5. Build Algorithms

6. Compare different Hypothesis fits of different algos

Section 1 –Libraries used

PANDAS

MATPLOTLIB

SKLEARN

NUMPY

Note : - you can install them using pip install <libraby name>

Section 2 – Dataset used in code

Iris flower dataset

Download from here –

https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv

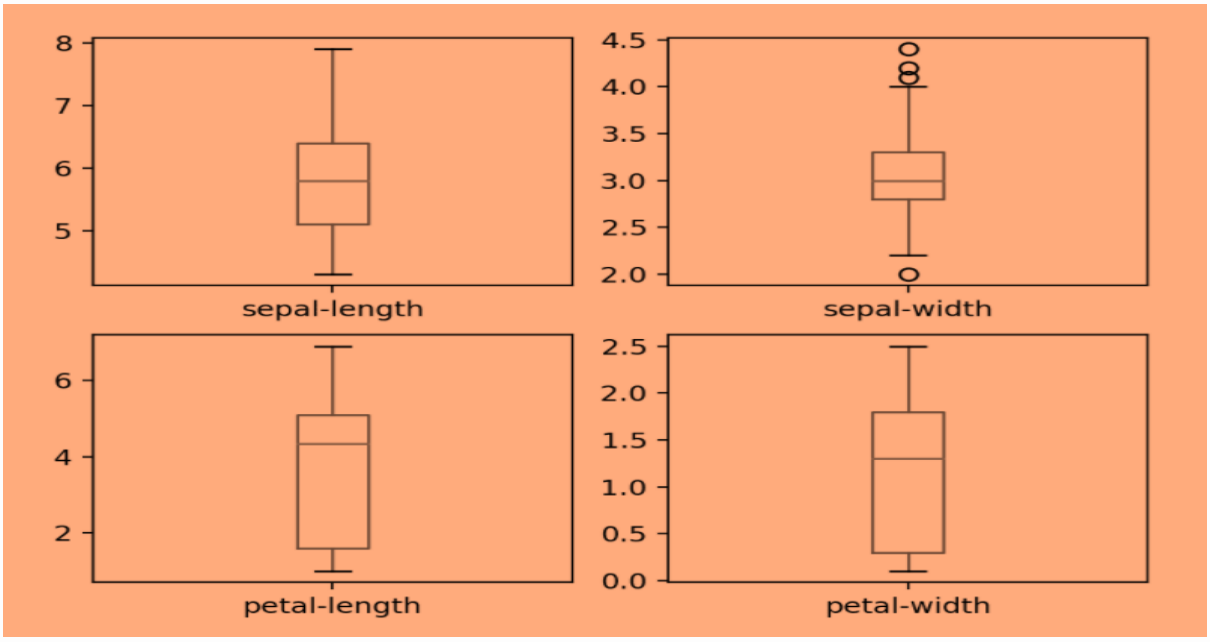
Section 3 - Dataset walk Through

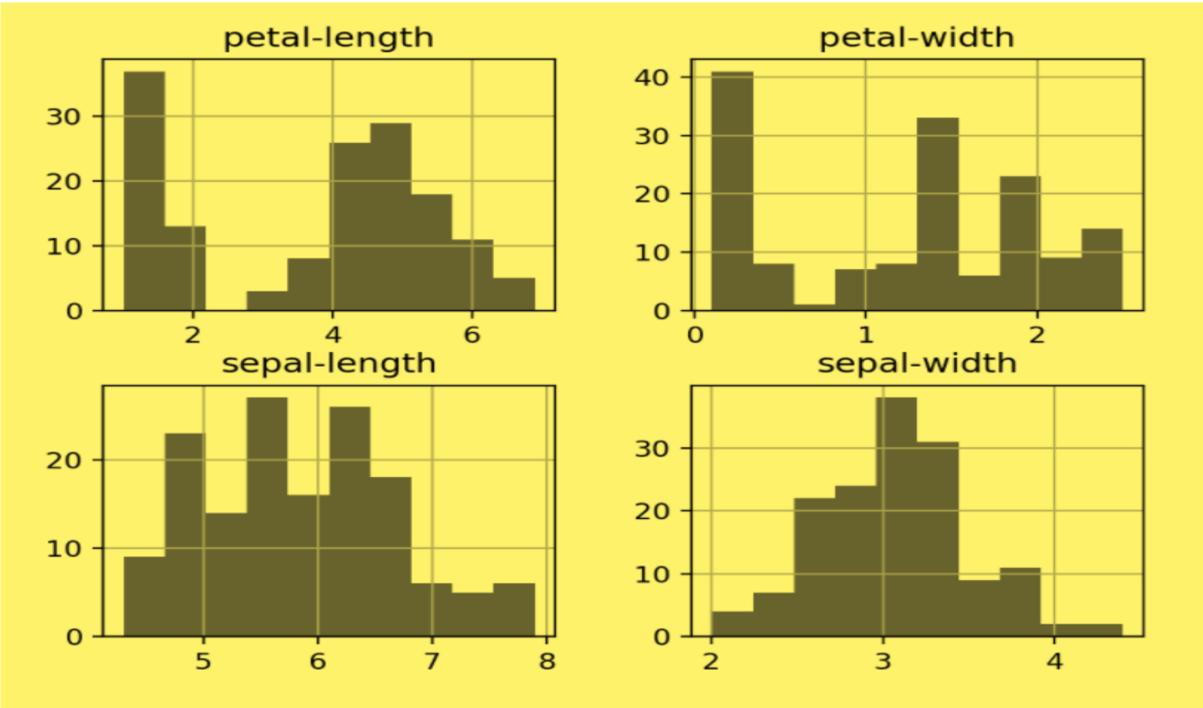
In the imported IRIS dataset we have m = 150 , and no of features = 5, so x=5

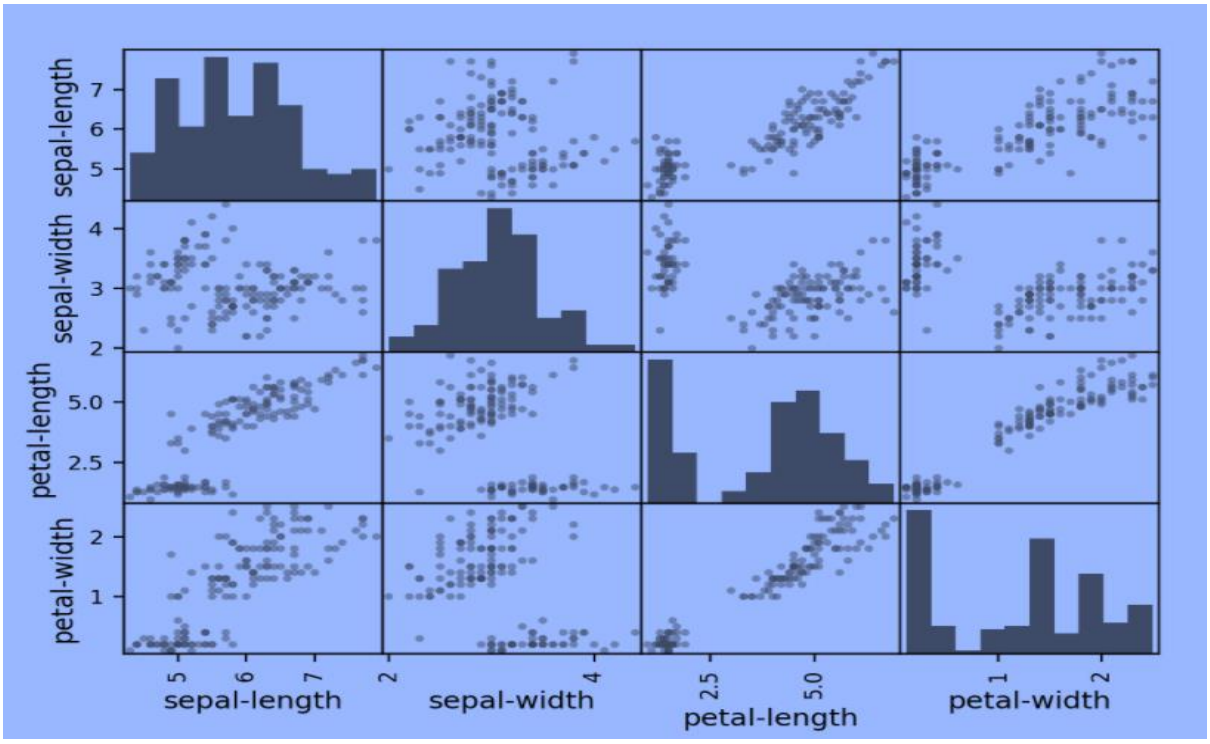
Feature Scaling is not required as all features are of same scale-

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Section – 4 – Dataset through Graphs







Section 5 – Build Algorithms

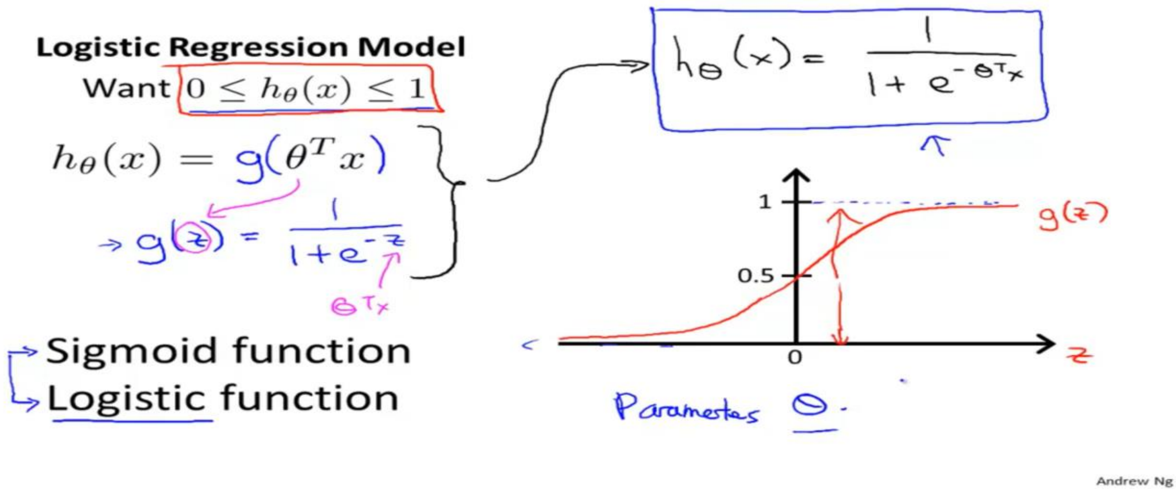
MY own Notes on Logistic regression-

Learning from – Stanford ML course by Andrew NG

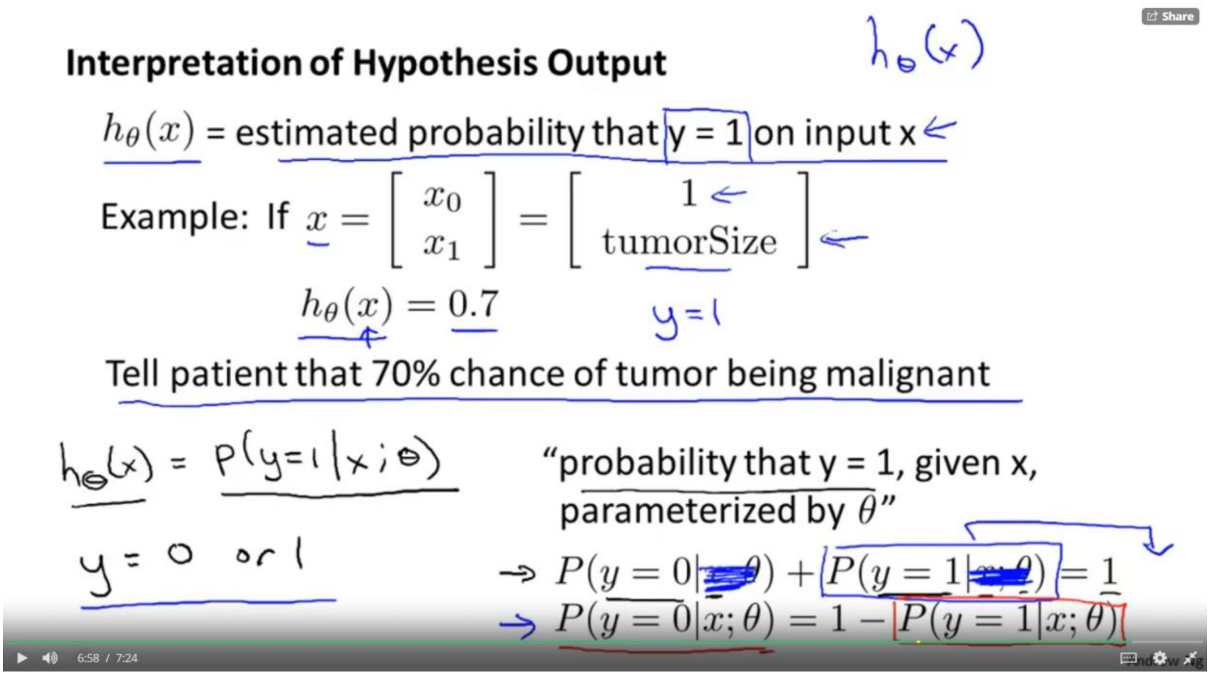
Logistic regression

Sigmoid or logistic function

So here we define our hypothesis in term so theta*x, but instead so just theta*x as in linear regression, we here make a little more involved function that we call as sigmoid function as it is given as in below slide, so our hypothesis is thus outputting between zero and 1 due to use of this sigmoid function and we treat this output as probability, thus if our hypothesis outputs 1 we say we 100 probability that tumor is benign



As in below example we chose x as 1 and tumor size, and our hypothesis gives value that we treat as probability

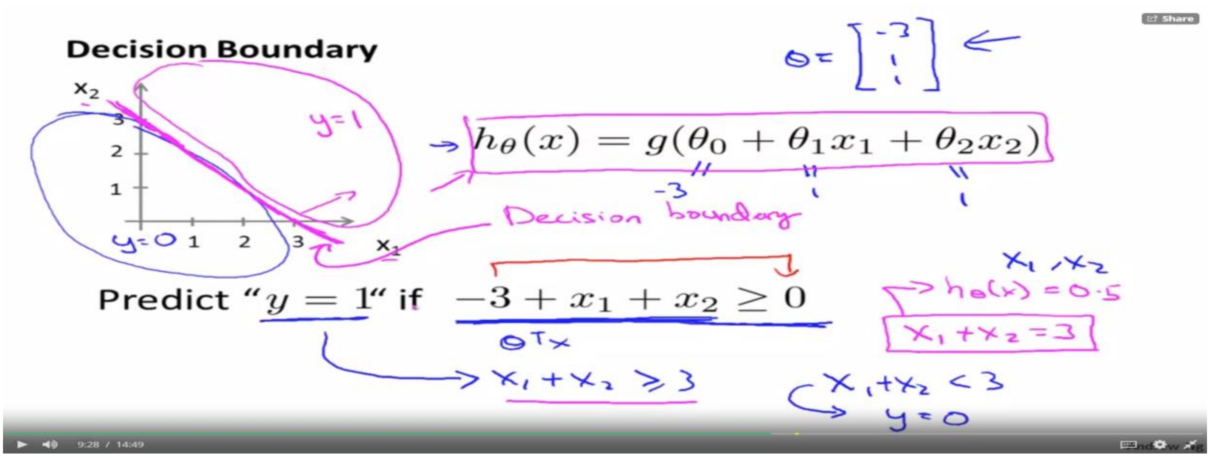


DECISION BOUNDARIES

These are key terms , these are used to demarcate boundaries from where you can say y is 1 or zero,

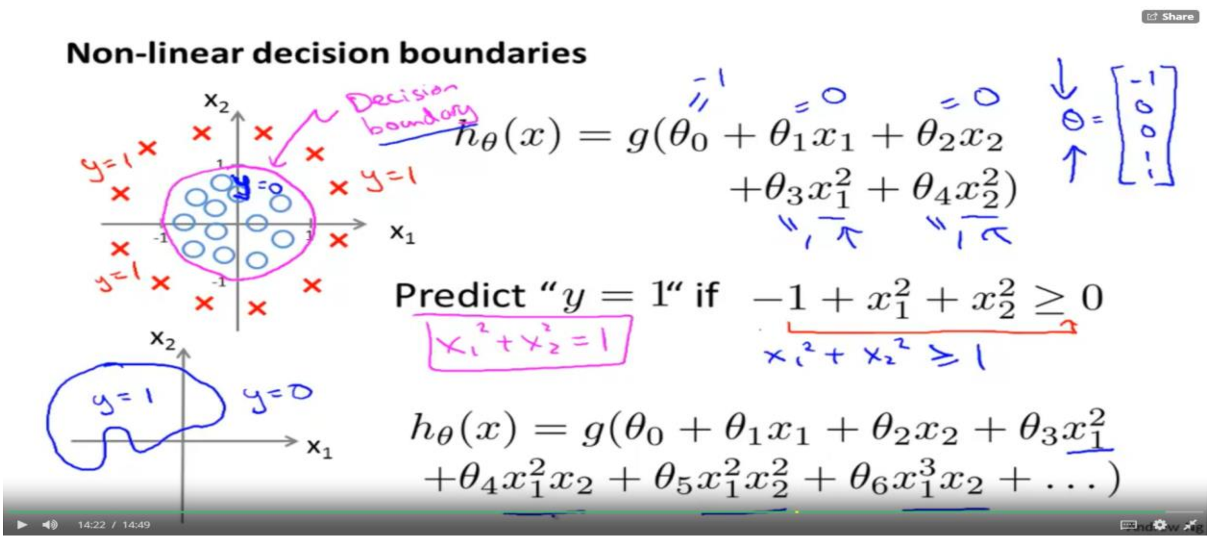
Also if you see sigmoid function , the value of that function is greater then 0.5 when z >0 and z is positive when theta*x is positive , so whole problem now boils down to the fact that if theta*x is +ve then we take y=1 and if negative then y=0, so a wonderful way to find the solution to classification problem using sigmoid function property of lying between 0 to 1 and having clear demarcation point of y= 0.5 at z=0

You can see in below example , if x1+ x2 >3 then we take y=1 otherwise y=0

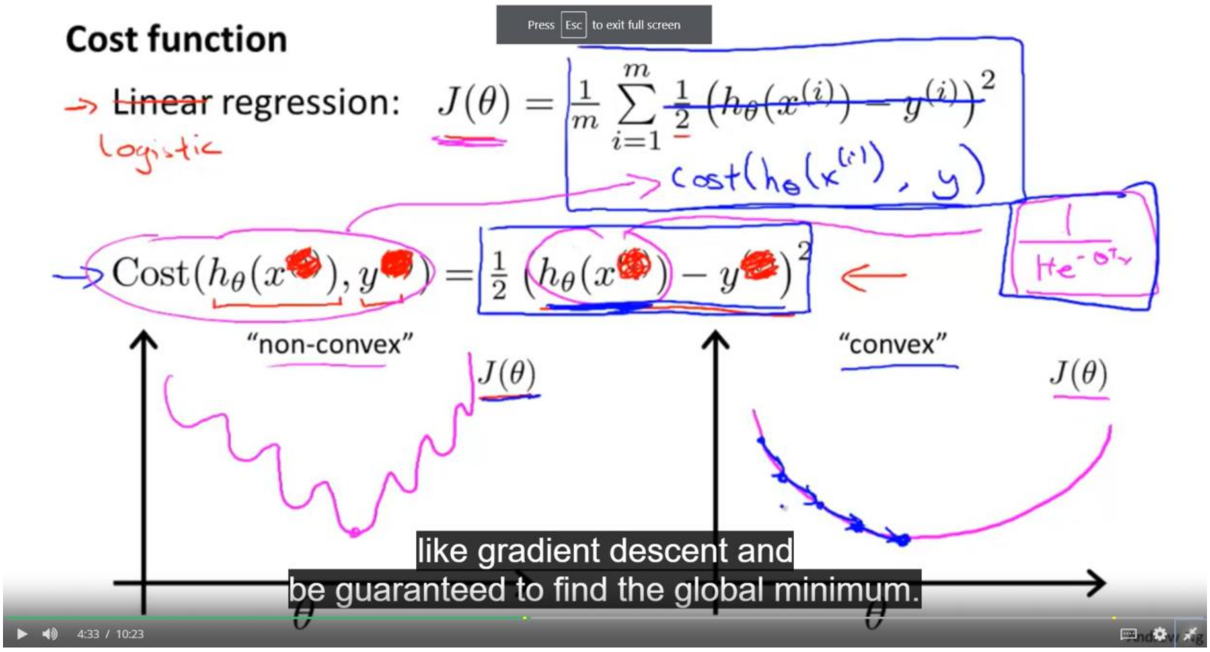


More complex clusters can be classified if you make features like below , so x1^2 and x2^2 are features, thus using these features gives power to cluster things inside and outside circles

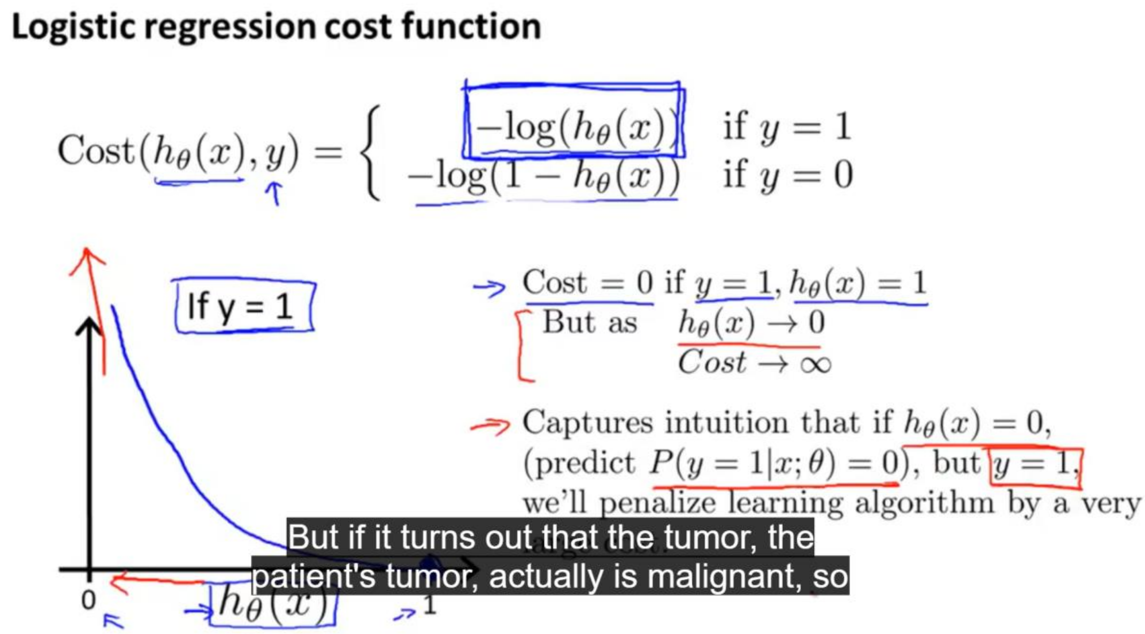
Also if you use features like x^2*y^2 ,you can even classify clusters of random shape, so in a way you can see galaxies can be clustered using similar algo, the features might be frequency of light coming .



How to choose cost function for a logistic regression problem



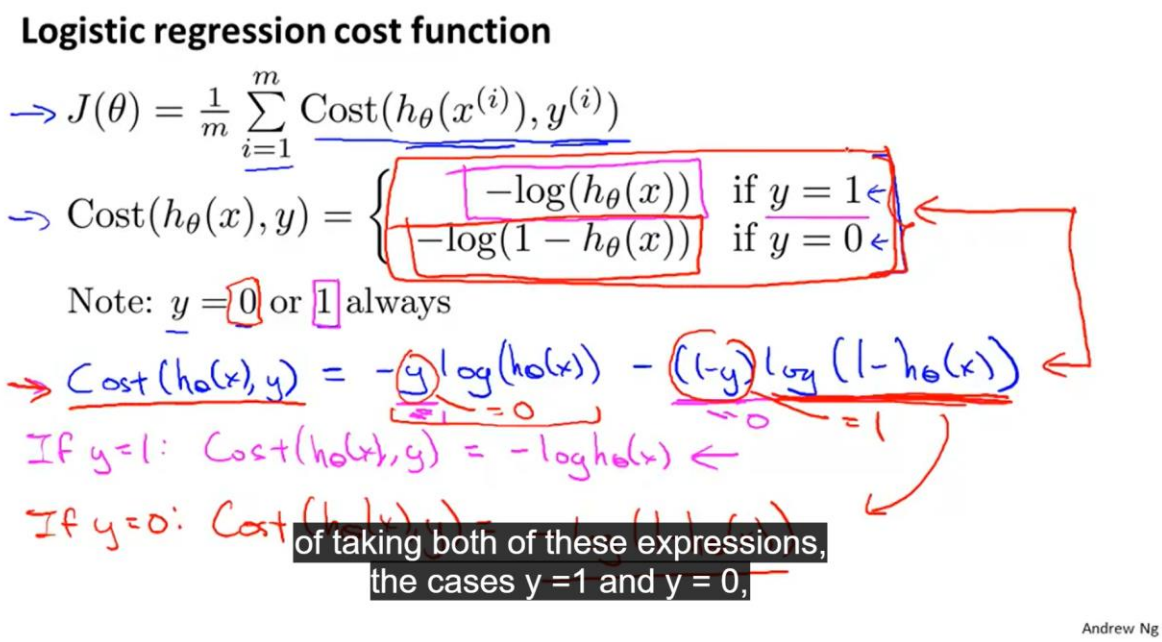
Now you can choose a cost function something like we have chosen in linear regression, but it turns out if use use that square valued function here , you get in to a function which is non convex in nature , i.e it might have many local minimas, so we have to pick a cost function that is convex in nature and ensures global minima



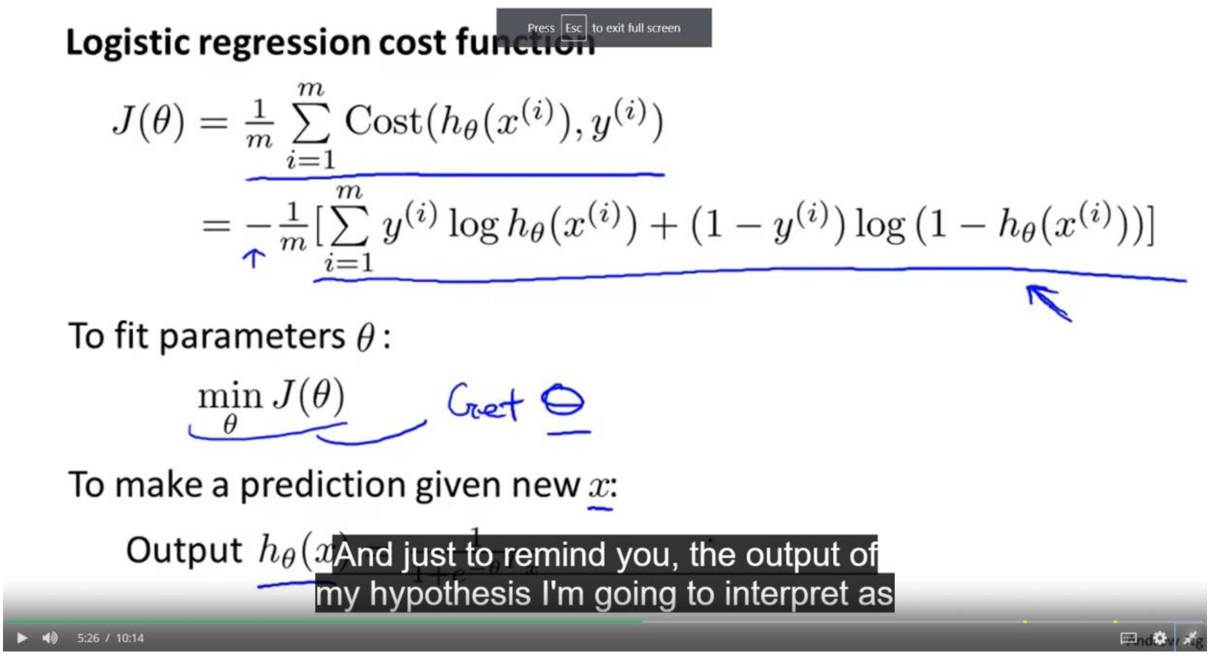
So we pick above cost function , now our choice of hypothesis ensures that we get values between 0 and 1 , so h(x) is always less 1, so we define our cost function as c = -log(h(x)) when y =1, remember y is coming from our training set, and h(x) is our hypothesis , so if you see -log(h(x)) you get 0 if h(x) is 1 , that means cost is 0 if your hypothesis predicts h(x) =1, so nice and if hypothesis predicts 0 but y =1 then cost is infinite , so we see our cost function working intuitively.

Applying gradient descent on logistic regression

Writing cost function in on line

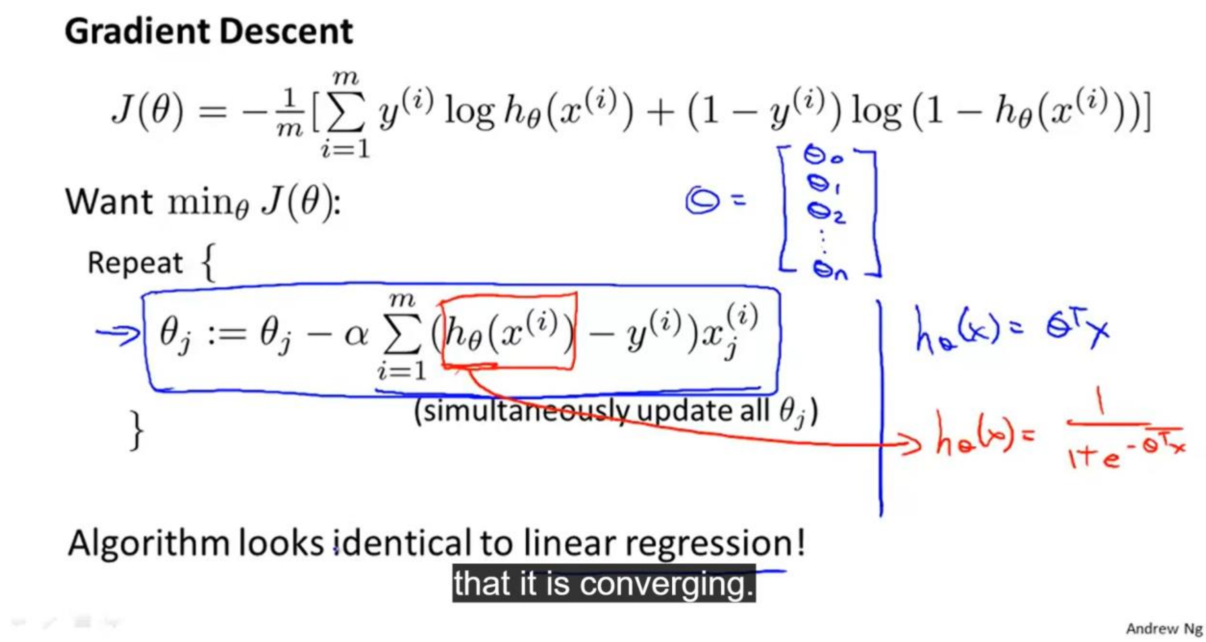


So our cost function looks like below, we train over all the set so summation sign and we want to minimise the cost of it, so all cool



Gradient descent, so same formula of parameter updating is applied in logistic regression as linear regression , what's different between the 2 is choice of hypothesis

You can derive for d/dtheta of current cost function it comes same as linear regression



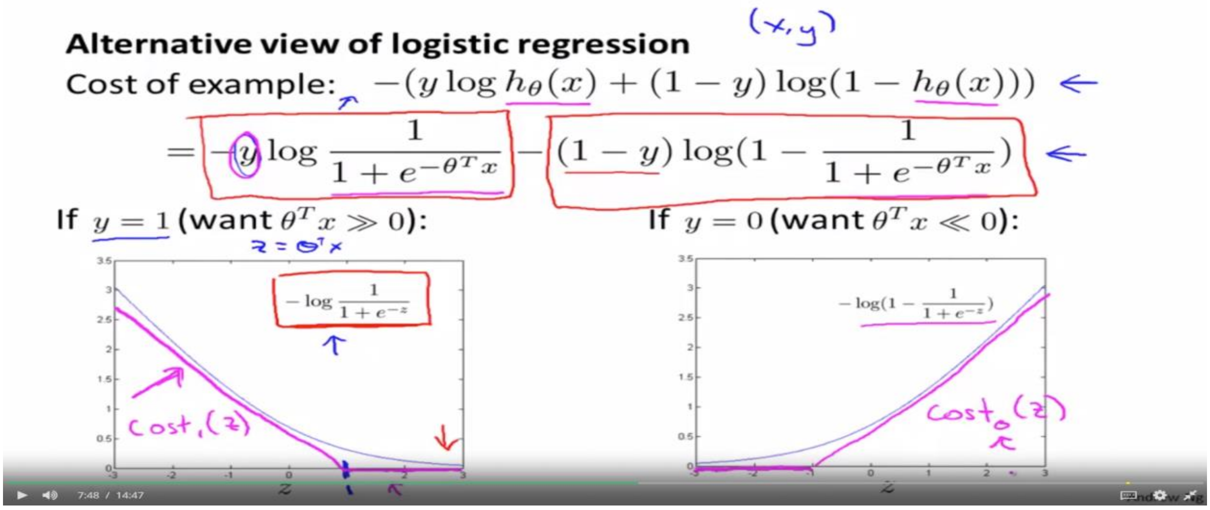
SVM algo –

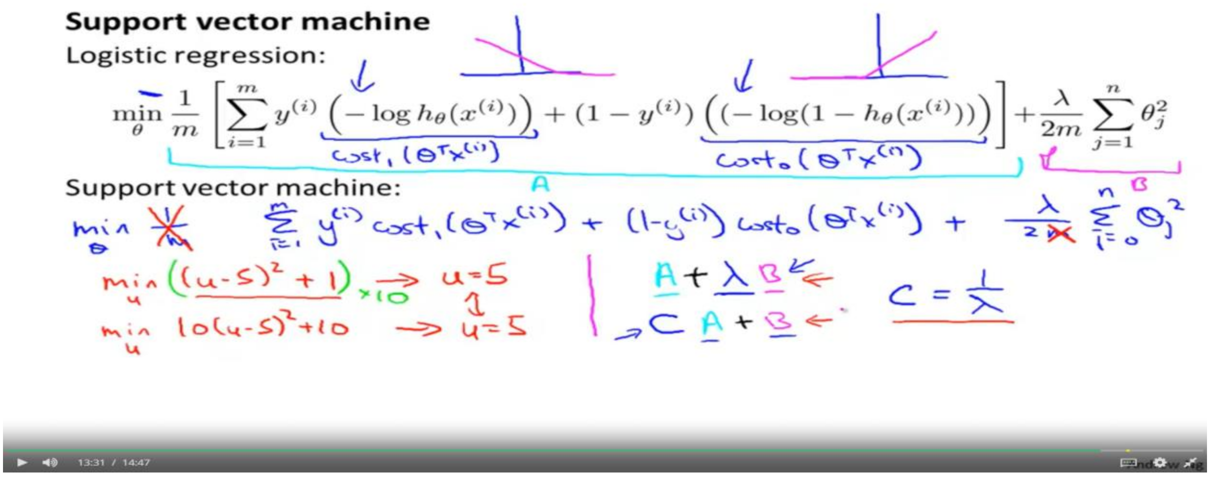
Support Vector Machine

One of the most used algos in the world, svm hypothesis is similar to logistic regression,

You can see below the cost function and hypothesis for logistic regression, the the regularisation term (lambda/2m)*sum(theta_j) is there to make sure we get smaller values of theta

In svm we just choose different hypothesis, these hypothesis are approximations of hypothesis of logistic regression for case y=1 and 0, also in cot function we drop m from denominator(svm people just prefer dropping it), also instead of having lambda in front of regularsation term we have C in front of hypothesis minmising term, this C act as 1/lambda, so giving same effect as regularisation.

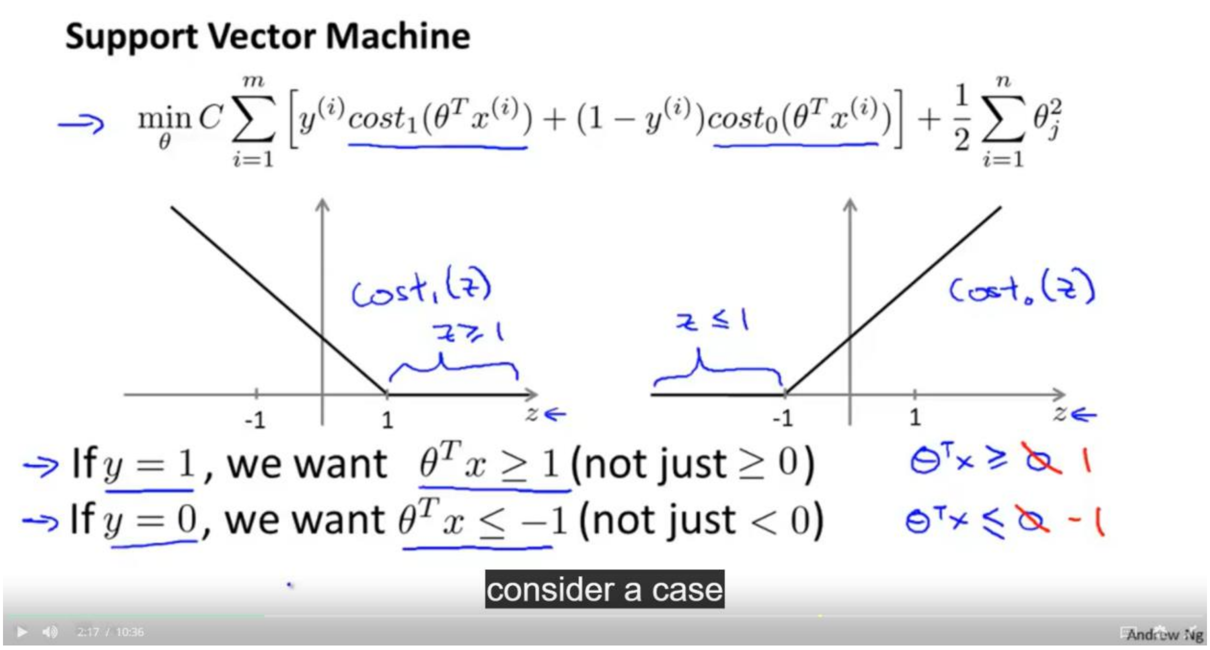




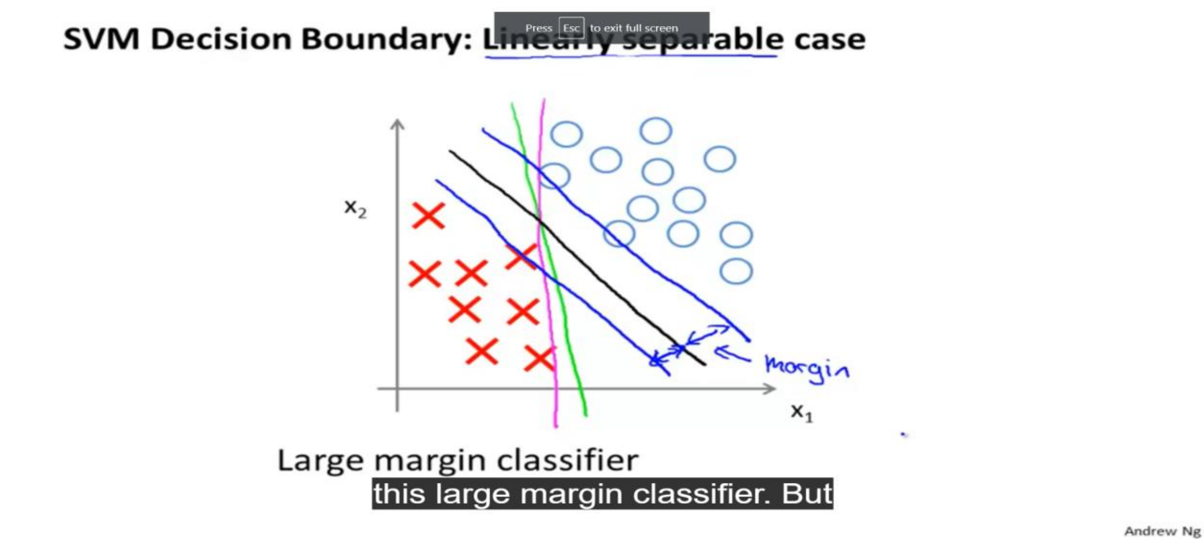
More Intuition on SVM

,

So you can think of a case in SVM when C is very large, so in this case what will happen is that in order to minimise J, the first term of cot function has to go to zero, and that hypothesis term goes to zero if we have theta*x <-1 if case of y=0 and theta*x >1 is case of y =1, so



The Svm algo due to its nature of choosing theta*x >1 , where theta*x>0 also would have worked , makes sure that we get a very distinctive fit , so this means we will get more separation or margin between 0 and 1 output, thus svm is called large margin classifier



SVM intuition via geometry

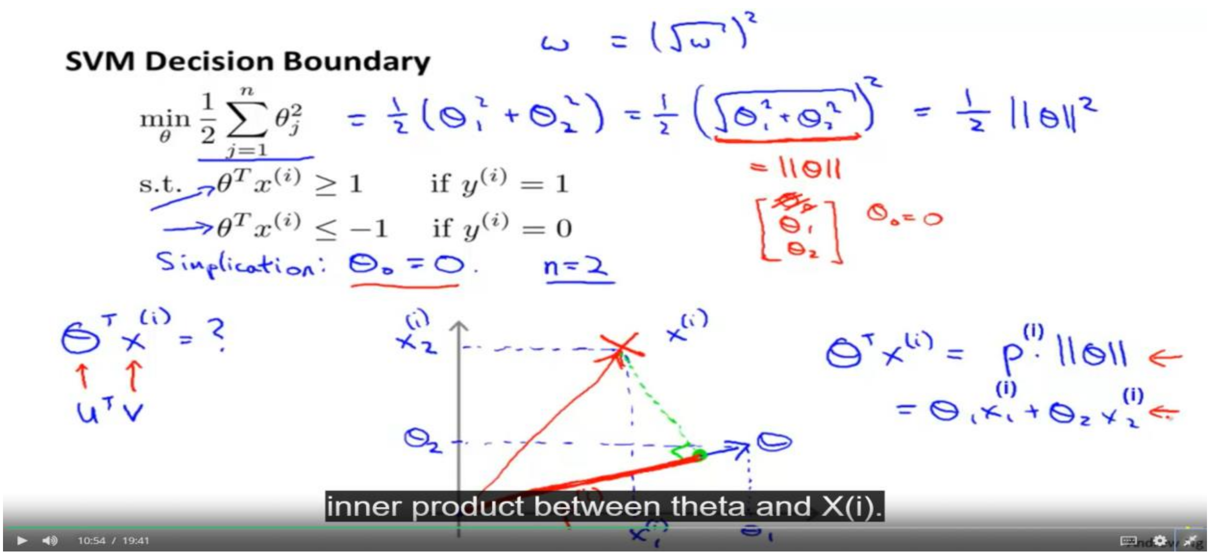
If you look closely for cases when C is very large we get to a cost function that is minimisation of (½)*sum(theta_j), keeping this mind we also see that,

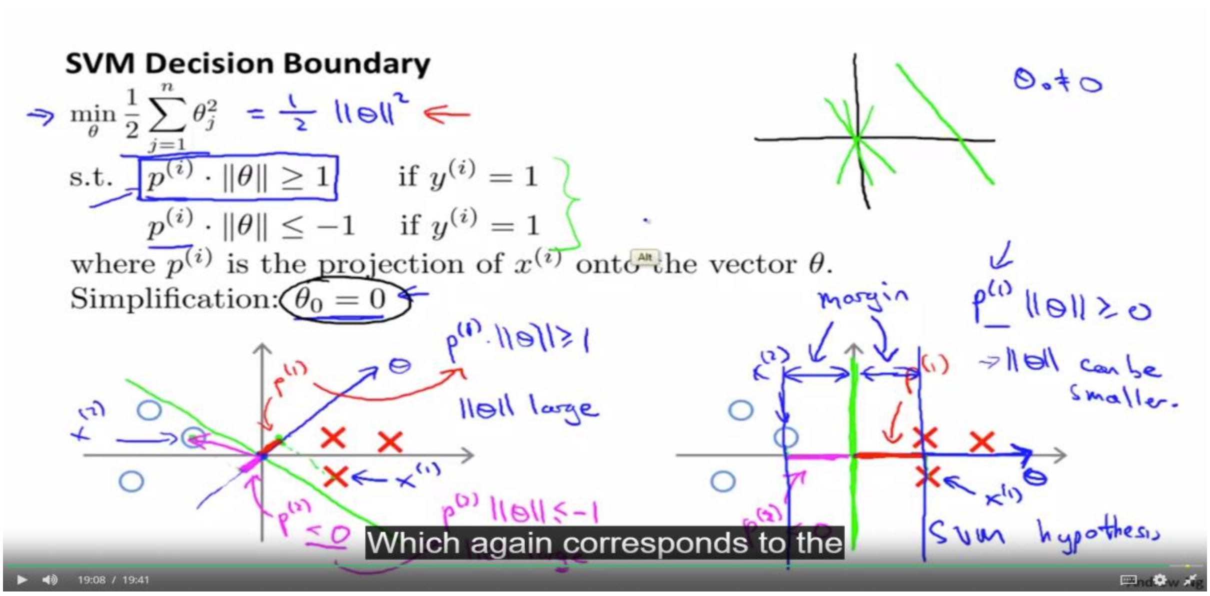
Theta*x is nothing but dot product of theta vector and x vector, and if look closely then,

Theta*x = p* mod(theta), where p is projection of x vector on theta vector which is as you have studied in jee is p = x.theta(unit vector) = x.theta/mod(theta)

So now you can see the power of this projection p , now we know that theta*x >1 for y=1, so that means p * mod(theta) > 1, but suppose if p is small then this would mean that theta has to be large for inequality to satisfy , but if theta large would mean increase of cost function , so this wouldn't happen , thus svm will try to keep P big, and making this p big ensures large margin between yes and no boundaries

So magnitude of P is like margin, greater of it ensures more margin , thus lesser theta which is what we want to minimise cost function un case when C>>>1





Kernels

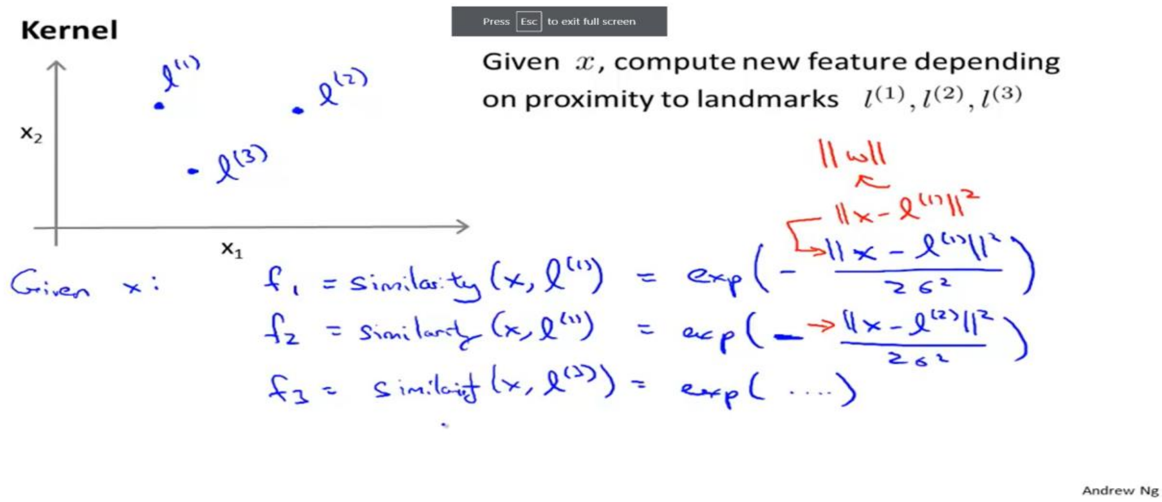
There was a question in logistic regression that if you have a non-linear decision boundary then you have increase your feature like cross features x1*x2e in order to fit the data, but as we realise in Computer vision that cross feature will substantially increase the numbers of features, so is there are features that are better than cross features and features that helps clasiification a great bit, it turns there is a mathematical way to define such features

We will call these features given by similalrity function , in this case we will use similarity function that are called gaussin kernels, kernels is synomous to similarity function,

So we pick some sampls from input set, i.e pick vector x ,cal them l1 , l2 , l3, l1 is set of feature , l2 is also set of feature for different input set.

Now e define gaussian kernel as f1 =exp(- (x-l)^2/2*sigma^2), so f1 is our new feature similarly, f2 and f3,

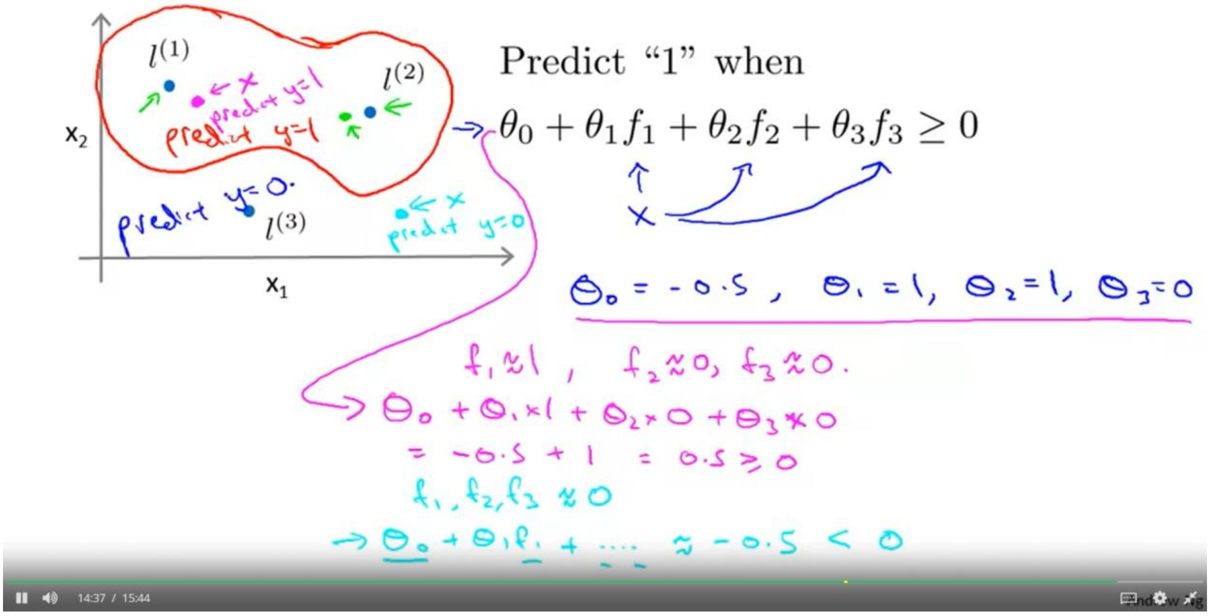
What that similarty function gives us is that , if x is close to l, then f1 = 1, otherwise if x is far away from l then f1 = 0



So we define decision boundary by theta_o + f1 *theta_1 + f2*theta_2 …

So if our new x point is far away from all l's , so we get f1 , f2 , f3 all zero and what we are left with is just theta_0, which is -ve thus theta*f <0, thus y=0

Also if say a point is close to l1 so all other theta*f will be zero but, theta_1*f1 will be 1, so -0.5 + 1 will positive so, y =1, thus this way we are able to predict y =1 for points close to l's and y=0 otherwsie



SVM cost function with kernels

One way to pick l"s or landmark is choose every x row as your landmark,so you will get n l's thus you will get n f features

X11 , x12 , x13 , y1

X21 , x22 , x23 , y2

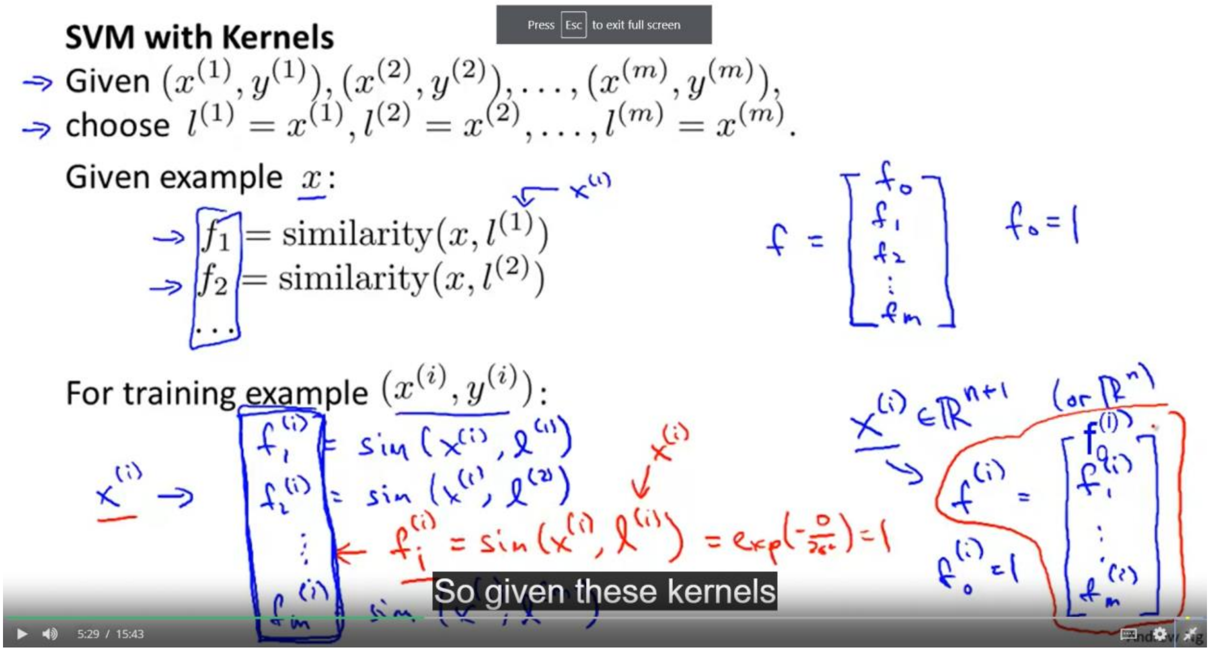
X31 , x32 , x33 , y3, old training set looks like this with x1,x2,3 as features for each row

F1 = f (x11-l1) + f(x12-l1) + f(x13-l1), y1

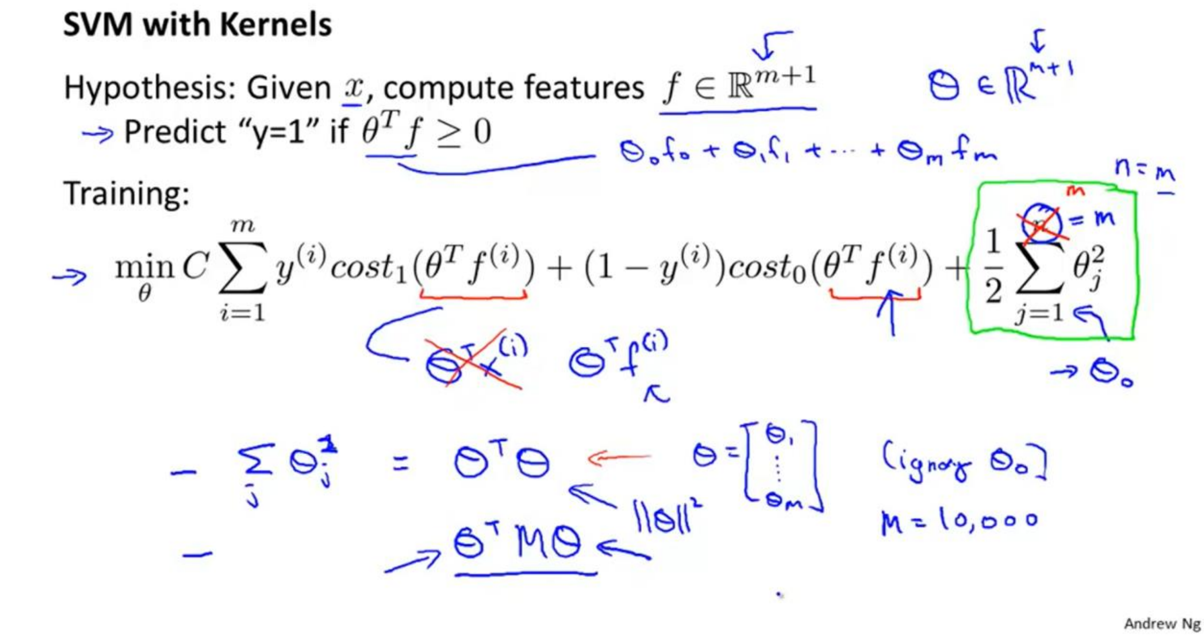
F 2 =f (x21-l1) + f(x22-l1) + f(x23-l1) , y2

F3 = f (x31-l1) + f(x32-l1) + f(x33-l1), y3 , so you have new training set like this, where x1 , x2 feature will be replaced by above

Sp your hypothesis becomes = y* cost_1 (theta*f) + (1-y)*cost_0(theta*f), so all the rows in dataset are replaced by single f_i feature



Sometimes svm people just choose to write regularisation term as theta_j^2 = theta_t *M* theta, where M is used to control somethings, that author didn't describe much, but author told not to worry abt M,

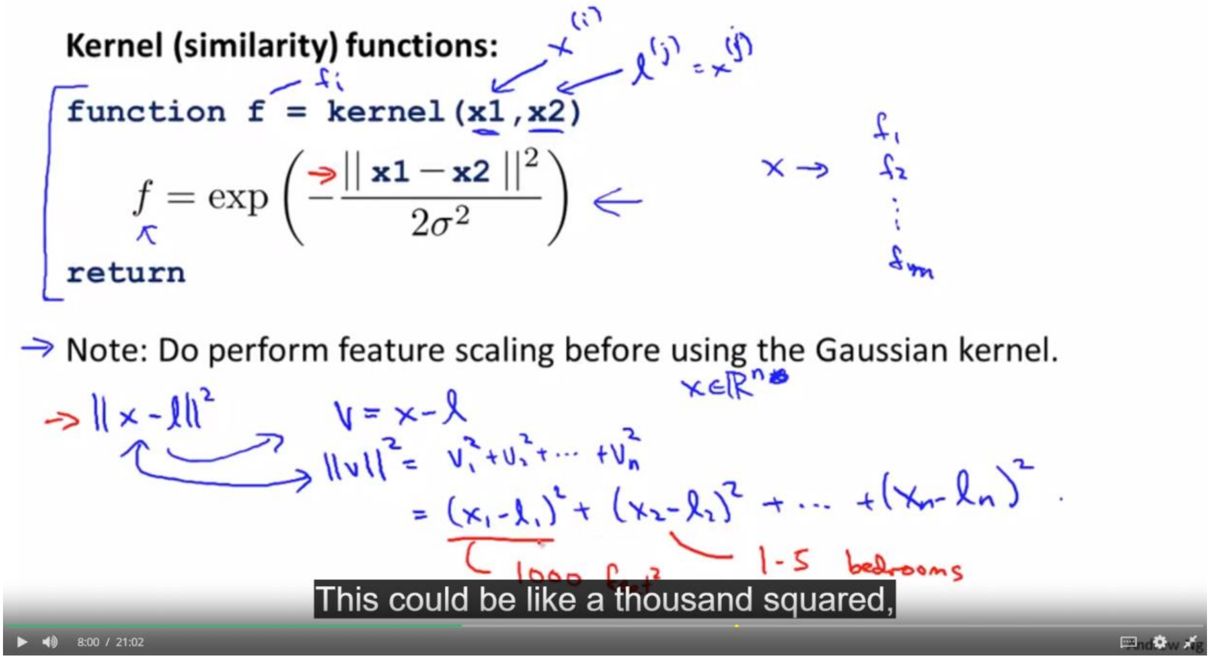


Also author told that off the shelf softwares are there to minismise above cost function

Final implementation of SVM

Solve cost function of svm by already available libraries

DO perform feature scaling when finding f1, f2 and so on



Logistic regression vs. SVMs

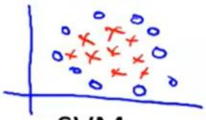
$n =$ number of features ($x \in \mathbb{R}^{n+1}$), $m =$ number of training examples

→ If n is large (relative to m): (E.g. $n \geq m$, $n = 10{,}000$, $m = 10 \ldots 1000$)

→ Use logistic regression, or SVM without a kernel ("linear kernel")

→ If n is small, m is intermediate: ($n = 1 - 1000$, $m = 10 - 10{,}000$) ←

→ Use SVM with Gaussian kernel



If n is small, m is large: ($n = 1 - 1000$, $m = 50{,}000+$)

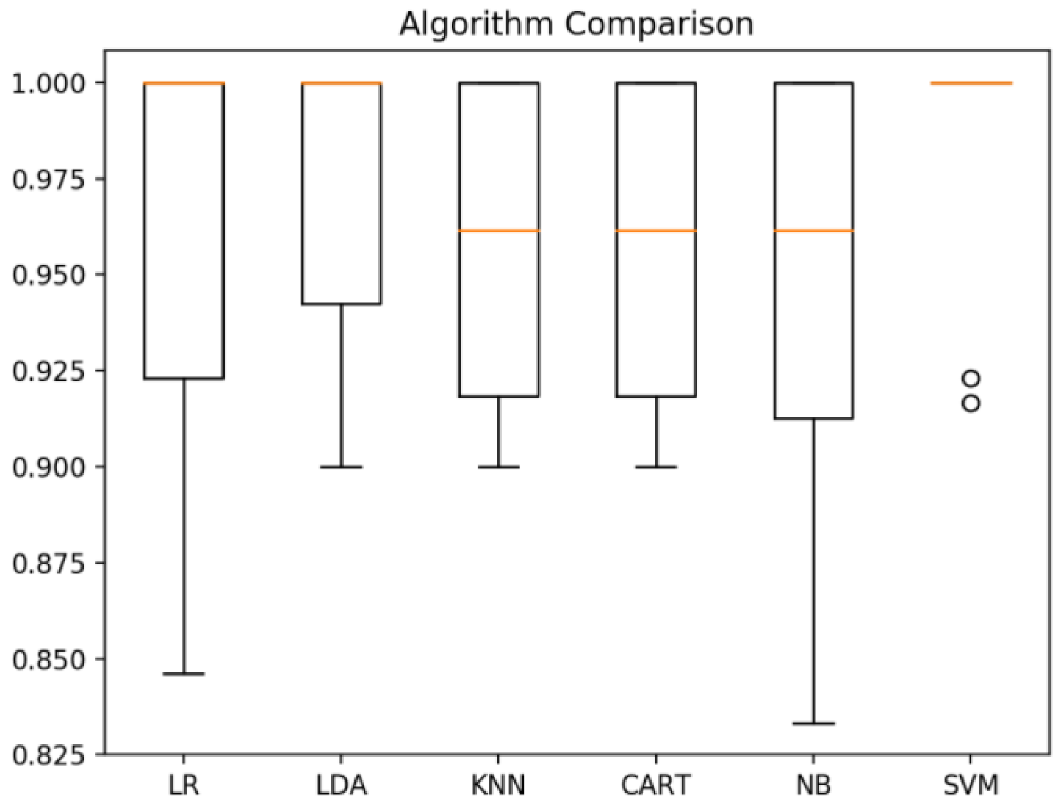
→ Create/add more features, then use logistic regression or SVM without a kernel

→ Neural network likely to work well for most of these settings, but may be slower to train.

The one disadvantage, or the one

Section 6 – Algorithm Comparison

Winner – Support Vector Machine Algorithm, as it showed greatest accuracy



Note : not the final documentation of code, will refine on it.